เครื่องมือสร้างภาพมโนทัศน์เพื่อสนับสนุนการออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิต

นายกานต์ ปุริสชาติ

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2549

A VISUALIZATION TOOL FOR SUPPORTING DIGIT-WISE ON-LINE COMPUTATIONAL DESIGN

Mr. Karn Purisachart

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Computer Science

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2006

หัวข้อวิทยานิพนธ์	เครื่องมือสร้างภาพมโนทัศน์เพื่อสนับสนุนการออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิต
โดย	นายกานต์ ปุริสชาติ
สาขาวิชา	วิทยาศาสตร์คอมพิวเตอร์
อาจารย์ที่ปรึกษา	อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์
อาจารย์ที่ปรึกษา (ร่วม)	อาจารย์ ดร.พิษณุ คนองชัยยศ

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

..... คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร.ดิเรก ลาวัณย์ศิริ)

คณะกรรมการสอบวิทยานิพนธ์

..... ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ)

..... อาจารย์ที่ปรึกษา
(อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์)

..... อาจารย์ที่ปรึกษาร่วม
(อาจารย์ ดร.พิษณุ คนองชัยยศ)

..... กรรมการ
(อาจารย์เชษฐ พัฒโนทัย)

กานต์ ปุริสชาติ : เครื่องมือสร้างภาพมโนทัศน์เพื่อสนับสนุนการออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิต (A VISUALIZATION TOOL FOR SUPPORTING DIGIT-WISE ON-LINE COMPUTATIONAL DESIGN) อ.ที่ปรึกษา: อ.ดร.อรรถสิทธิ์ สุรฤกษ์, อ.ที่ปรึกษาร่วม: อ.ดร.พิษณุ คนองชัยยศ 192 หน้า.

วิทยานิพนธ์นี้มีวัตถุประสงค์เพื่อพัฒนาเครื่องมือสร้างภาพมโนทัศน์ เพื่อสนับสนุนการออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิต เครื่องมือที่ได้ช่วยให้นักออกแบบประหยัดเวลาในการออกแบบและการวางแผนผังของการประกอบหน่วยย่อยของการคำนวณ ตลอดจนสามารถเห็นการเปลี่ยนแปลงสถานะการทำงานของอัลกอริทึมได้

วิทยานิพนธ์ฉบับนี้เสนอเครื่องมือที่สามารถแปลงอัลกอริทึมภาษาโปรแกรม และแสดงออกเป็นภาพเคลื่อนไหวที่แสดงถึงการไหลของข้อมูลระดับดิจิตผ่านไปตามคำสั่งต่างๆ ซึ่งช่วยทำให้นักออกแบบสามารถเห็นสถานะการส่งค่าในระดับดิจิตได้อย่างชัดเจน

เครื่องมือนี้ได้รับการออกแบบและพัฒนาโดยใช้แนวคิดการออกแบบภาษาสำหรับใช้แสดงภาพมโนทัศน์และสร้างส่วนประมวลผลภาพมโนทัศน์ โดยแสดงการทำงานผ่านส่วนประสานงานกับผู้ใช้ เครื่องมือสามารถสร้างหน่วยย่อยของการคำนวณและนำกลับมาใช้ใหม่ได้

จากการทดสอบการทำงานของเครื่องมือทั้งหมดพบว่าเครื่องมือนี้สามารถทำงานได้ถูกต้อง ตามที่ได้ออกแบบไว้

ภาควิชา ...วิศวกรรมคอมพิวเตอร์.....ลายมือชื่อนิสิต.....
สาขาวิชา...วิทยาศาสตร์คอมพิวเตอร์..ลายมือชื่ออาจารย์ที่ปรึกษา.....
ปีการศึกษา.....2549.....ลายมือชื่ออาจารย์ที่ปรึกษา (ร่วม).....

4671402021 : MAJOR COMPUTER SCIENCE

KEY WORD: VISUALIZATION ON-LINE DIGIT-WISE / REDUNDANT NUMBER

KAN PURISACHART: A VISUALIZATION TOOL FOR SUPPORTING DIGIT-WISE ON-LINE COMPUTATIONAL DESIGN. THESIS ADVISOR: ATHASIT SURARERKS, Ph.D., THESIS CO-ADVISOR: PIZZANU KANONGCHAIYOS, Ph.D., 192 pp.

The objective of this thesis is to develop a visualization tool for supporting digit-wise on-line computational design. This tool will help a designer save time on designing and laying out a component computation unit, and enable him to observe changes in data during the algorithm process.

This thesis presents a tool that converts program-language algorithms and then displays graphically the flow of digit-level data through each command. This allows the designer to clearly see the digit value in every state.

This tool was designed and developed by using a programming language concept to interface visualization and create visualization engines to present the process through a user interface. Additionally, it can create computation units and reuse them.

This software tool is tested. The results show that this software performs correctly as designed.

Department :...Computer Engineering.....Student's Signature :.....Kan Purisachart.....

Field of Study :....Computer Science.....Advisor's Signature :.....Athasit Surarerks.....

Academic Year :.....2006.....Co-advisor's Signature :.....

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยความเมตตากรุณาอย่างยิ่งของ อาจารย์ ดร.อรรถสิทธิ์ สุรฤกษ์ ขอกราบขอบพระคุณอาจารย์ที่กรุณาสละเวลา ตรวจทานแก้ไขให้คำแนะนำ คำปรึกษา และความช่วยเหลือ ขอกราบขอบพระคุณ อาจารย์ ดร.พิษณุ คนองชัยยศ อาจารย์ที่ปรึกษา(ร่วม) ที่กรุณาให้คำแนะนำ รวมทั้งกรุณาช่วยตรวจทาน แก้ไขเนื้อหาวิทยานิพนธ์ ผู้เขียนรู้สึกซาบซึ้งในความเมตตาของท่านเป็นอย่างมาก และขอกราบขอบพระขอบคุณคณะกรรมทุกท่าน ผู้ช่วยศาสตราจารย์ ดร.วิวัฒน์ วัฒนาวุฒิ และอาจารย์เชษฐ พัฒโนทัย ที่กรุณาสละเวลาให้คำแนะนำ และตรวจทานแก้ไขวิทยานิพนธ์ และขอกราบขอบพระคุณอาจารย์ทุกท่านที่ได้ให้ความรู้ และความเมตตาต่อข้าพเจ้าในระหว่างการศึกษา

ขอขอบคุณ คุณดลนภา บุญญาพงศ์นุกูล ที่เป็นกำลังใจ และให้ความช่วยเหลือในทุกๆ ด้านแก่ผู้เขียนเพื่อสนับสนุนการทำวิทยานิพนธ์นี้ให้สำเร็จได้ด้วยดี ขอขอบคุณ เพื่อนๆ พี่ๆ น้องๆ เพื่อนป.ตรี เพื่อนป.โท และที่ทำงาน ที่ให้ความช่วยเหลือ ให้คำปรึกษา ให้กำลังใจและมีน้ำใจกับข้าพเจ้าตลอดมา

ท้ายนี้ หากวิทยานิพนธ์ฉบับนี้มีคุณค่าและประโยชน์ประการใดแล้ว ผู้เขียนขอกราบเป็นกตเวทิตาคุณแก่บิดามารดา คณาจารย์ และผู้มีพระคุณทุกท่านของผู้เขียน

สารบัญ

บทที่ หน้า

สารบัญตาราง

ตาราง หน้า

สารบัญภาพ

ภาพประกอบ หน้า

ภาพประกอบ หน้า

ภาพประกอบ หน้า

บทที่ 1

บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ศาสตร์ในด้านการออกแบบระบบการคำนวณให้กับคอมพิวเตอร์ เป็นศาสตร์ที่มีความสำคัญมากในการพัฒนาประสิทธิภาพของคอมพิวเตอร์ ประสิทธิภาพในที่นี้ หมายถึงประสิทธิภาพในด้านการคำนวณ เช่น ความเร็วในการคำนวณ การประหยัดเนื้อที่ในการคำนวณ และขนาดของวงจรมีขนาดเล็ก เป็นต้น ในการเพิ่มประสิทธิภาพในด้านการคำนวณให้กับคอมพิวเตอร์นั้น ระบบจำนวนเป็นปัจจัยสำคัญเช่นกัน ปัจจุบันระบบจำนวนแบบฐานสองเป็นที่ยอมรับ และใช้อย่างแพร่หลายในคอมพิวเตอร์ เพราะรูปแบบของจำนวนสอดคล้องกับลักษณะการทำงานของคอมพิวเตอร์ แต่การคำนวณต่างๆ บนระบบจำนวนฐานสองก็มีข้อจำกัดมากมาย เช่น ไม่สามารถคำนวณได้ด้วยวิธีการคำนวณความเร็วสูง หรือการคำนวณแบบขนาน (Parallel computation) ดังนั้นในการพัฒนาระบบการคำนวณของคอมพิวเตอร์ จึงได้มีนักวิจัยออกแบบระบบจำนวนที่มีประสิทธิภาพสูงในด้านความเร็วในการคำนวณ พร้อมทั้งออกแบบวิธีการคำนวณบนระบบนั้นด้วย อันได้แก่ ระบบจำนวนแบบมีเครื่องหมายของอเวเซียนีส (Avizienis's Signed Digit Number System) [1] โดยระบบนี้เป็นระบบจำนวนแบบซ้ำซ้อน (Redundant Number System) สามารถคำนวณแบบขนานในระบบนี้ได้ นอกจากนี้รูปแบบของการคำนวณก็เป็นอีกปัจจัยในการเพิ่มประสิทธิภาพ เช่น การคำนวณแบบเชื่อมตรงและท่อตรง (On-line and Pipeline Computation) เป็นการนำการคำนวณแบบลำดับ (Sequential) มาทำงานต่อเนื่องกันแบบท่อตรง ทำให้เวลาในการคำนวณเทียบเท่ากับการคำนวณแบบขนาน อัลกอริทึมการคำนวณพื้นฐาน เช่น การบวก การลบ การคูณ และการหาร มีความซับซ้อนมากขึ้นจากระบบจำนวนแบบปกติ ทำให้ยากในการทำความเข้าใจสำหรับผู้ออกแบบอัลกอริทึม

ปัญหาในการออกแบบอัลกอริทึมการคำนวณแบบเชื่อมตรงอย่างหนึ่งคือ การนำอัลกอริทึมที่ได้ไปใช้ในการทำให้เป็นจริง (Implementation) ซึ่งหมายถึง การนำอัลกอริทึมไปสร้างวงจรการคำนวณ โดยปกติแล้วการคำนวณแบบเชื่อมตรงจะทำงานโดยรับข้อมูลนำเข้ามาทีละหลัก และผลิตคำตอบทีละหลักเช่นกัน การดำเนินการคำนวณจะทำไปตามลำดับ ส่วนที่สำคัญสำหรับอัลกอริทึมคือ มีลักษณะของการส่งค่าระหว่างสถานะ (สถานะในที่นี้หมายถึง สถานะที่เปลี่ยนไปตามข้อมูลนำเข้าที่ผ่านมาทั้งหมด) ดังนั้นสมการการคำนวณในอัลกอริทึมจึงมีลักษณะของการวนซ้ำ และการส่งค่าของตัวแปรวนซ้ำเป็นส่วนใหญ่ และลักษณะเช่นนี้จะมีการเกิดขึ้นใน

หลายจุดการคำนวณ แต่จากปัญหาการวนซ้ำจึงทำให้การออกแบบแผนผังการคำนวณมีความซับซ้อนสูง ดังนั้นเครื่องมือที่ช่วยในการออกแบบจึงต้องมีความสามารถในการแปลงอัลกอริทึมจากที่เขียนอยู่ในลักษณะโปรแกรมการทำงานตามลำดับของคำสั่งหรือลำดับของการคำนวณ ที่ทำให้เห็นการส่งค่าในระดับดิจิตได้ชัดเจน โดยแสดงเป็นแผนภาพที่แสดงถึงการไหลของข้อมูลระดับดิจิตที่ผ่านไปตามคำสั่งต่างๆ ได้

ในงานวิทยานิพนธ์นี้สนใจการสร้างเครื่องมือช่วยให้นักออกแบบประหยัดเวลาในการวางแผนผังของการประกอบหรือประหยัดเวลาในการออกแบบหน่วยย่อยของการคำนวณได้ โดยการจำลองอัลกอริทึมที่เขียนอยู่ในรูปแบบโปรแกรมการทำงานแสดงออกเป็นรูปแผนภาพเคลื่อนไหว ทั้งนี้เพื่อเป็นประโยชน์ในการช่วยให้เห็นการเปลี่ยนแปลงของสถานะการทำงาน ได้ตามต้องการ และหน่วยย่อยหรือส่วนประกอบย่อยที่ออกแบบนี้สามารถนำไปใช้ในการคำนวณแบบเชื่อมตรงแบบต่างๆ ได้เช่นกัน เครื่องมือที่สร้างจึงต้องมีลักษณะที่สามารถตอบสนองความต้องการนี้ได้

1.2 วัตถุประสงค์ของการวิจัย

ออกแบบและพัฒนาเครื่องมือสร้างภาพมโนทัศน์เพื่อสนับสนุนการออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิต

1.3 ขอบเขตของการวิจัย

1.3.1 ออกแบบและพัฒนาสร้างเครื่องมือสร้างภาพมโนทัศน์สำหรับการคำนวณแบบเชื่อมตรงระดับดิจิต

1.3.2 กลุ่มคำสั่งพื้นฐานที่ใช้ในการเขียนคำสั่งอัลกอริทึมประกอบด้วย

1.3.2.1 การทำงานของตัวแปรจะทำงานอยู่บนพื้นฐานของเลขฐานสองระดับดิจิต และต้องระบุตัวแปรที่เป็นตัวแปรนำเข้าและตัวแปรที่เป็นผลลัพธ์ให้ชัดเจน

1.3.2.2 คำสั่งกำหนดค่า เป็นคำสั่งกำหนดค่าให้กับตัวแปรต่างๆ ในอัลกอริทึม

1.3.2.3 คำสั่งควบคุมเส้นทางการทำงาน ใช้กับคำสั่งการทำงานแบบวนซ้ำ (While-loop) คำสั่งกำหนดเงื่อนไข (If) และคำสั่งเรียกโปรแกรมย่อย (Call)

1.3.2.4 ตัวดำเนินการที่ใช้กับตัวแปร ใช้ตัวดำเนินการระดับดิจิต และตัวดำเนินการสัมพันธ์ (Relational operator)

1.3.3 สามารถสร้างภาพมโนทัศน์การทำงานของอัลกอริทึมและแสดงผลในรูปแบบกราฟิกที่ผู้ใช้สามารถเห็นการเปลี่ยนแปลงของข้อมูลนำเข้าและผลลัพธ์ผ่านทางจอภาพ

1.4 ขั้นตอนการดำเนินงานวิจัย

1.4.1 ศึกษาและออกแบบการแปลคำสั่งอัลกอริทึม

1.4.2 ศึกษาอัลกอริทึมการคำนวณแบบเชื่อมตรง

1.4.3 ศึกษาวิธีการเขียนโปรแกรมในการสร้างเครื่องมือการคำนวณทางแบบเชื่อมตรงระดับดิจิต

1.4.4 ออกแบบและพัฒนาเครื่องมือสร้างภาพมโนทัศน์การคำนวณแบบเชื่อมตรงระดับดิจิต

1.4.5 ทดสอบเครื่องมือกับคำสั่งอัลกอริทึมการคำนวณแบบเชื่อมตรงระดับดิจิต

1.4.6 สรุปผลการวิจัย ข้อเสนอแนะ และจัดทำวิทยานิพนธ์ฉบับสมบูรณ์

1.5 ประโยชน์ที่คาดว่าจะได้รับ

1.5.1 ได้เครื่องมือสร้างภาพมโนทัศน์เพื่อสนับสนุนการออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิต

1.5.2 ทำให้ผู้ออกแบบเข้าใจขั้นตอนการทำงานของการคำนวณแบบเชื่อมตรงระดับดิจิตได้ชัดเจนและง่ายขึ้น

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

ทฤษฎีและงานวิจัยที่เกี่ยวข้องในงานวิจัยนี้ ประกอบด้วยระบบจำนวน และอัลกอริทึมการคำนวณทางคณิตศาสตร์แบบเชื่อมตรง และวิธีการเชิงเหตุการณ์ มีรายละเอียดดังนี้

2.1.1 ระบบจำนวน (Number System)

ระบบจำนวนเขียนอยู่ในรูปของ (β , D) ประกอบด้วยเลขฐาน β ที่สามารถเป็นได้ทั้งจำนวนจริงและจำนวนเชิงซ้อน โดยที่ β>1 และชุดตัวเลขแบบจำกัด (Finite Digit Set) D ที่ตัวเลขเป็นได้ทั้งจำนวนจริง และจำนวนเชิงซ้อน

ในระบบจำนวนตามวิธีแบบปกติ ถ้า β เป็นจำนวนเต็มและมีชุดตัวเลข D อยู่ในรูปของ $\{ c \in I \mid 0 \leq c \leq |\beta| - 1 \}$ เรียกชุดตัวเลขนี้ว่า ชุดตัวเลขแบบคาโนนิคอล (Canonical Digit Set) แทนด้วย C ซึ่ง C จะมีค่าอยู่ระหว่าง {0, 1, 2, 3, …, β-1} สำหรับระบบจำนวนที่ประกอบด้วยเลขฐาน β สามารถแทนตัวเลขของ X ใน D ที่มีเลขฐาน β อยู่ในรูปแบบดังนี้

$$X = (X_a X_{a-1} X_{a-2} X_{a-3} X_{a-4} \ldots X_0 . X_{-1} X_{-2} X_{-3} \ldots)_\beta \quad \text{โดยที่ } X_i \in D \text{ และ } i \leq a$$

ค่าเชิงตัวเลข (Numerical Value) ของ X ในฐาน β เขียนแทนด้วยสัญลักษณ์ ||X|| นิยามดังนี้ $||X|| = \sum_{i=a}^{-\infty} X^i \beta^i$

ในระบบจำนวน (β , D) ถ้าสามารถแทนตัวเลขที่มีเลขฐาน β ที่ยาวจำกัดได้ทั้ง X_1 และ X_2 โดยที่ $||X_1|| = ||X_2||$ แล้ว เรียกระบบจำนวนนี้ว่าระบบจำนวนซ้ำซ้อน (Redundant Number System) ตัวอย่างเช่น กำหนดให้ β = 2 และ D = {-1, 0, 1} ซึ่งสามารถเขียนค่า 5 ได้มากกว่าหนึ่งรูปแบบดังนี้ $(101)_2$ และ $(10\text{-}1\text{-}1)_2$ เป็นต้น

2.1.2 การคำนวณทางคณิตศาสตร์แบบเชื่อมตรง (Online Arithmetic Computation) [2]

ตัวดำเนินการพื้นฐานสำหรับการคำนวณแบบลำดับโดยทั่วไปเช่น การคูณ การบวก และการลบ โดยปกติแล้วจะทำงานเริ่มต้นที่หลักที่มีนัยสำคัญต่ำสุด (Least Significant Digit

First, LSDF) แต่การหารจะทำการในรูปแบบตรงกันข้ามคือ เริ่มต้นที่หลักที่มีนัยสำคัญสูงสุด (Most Significant Digit First, MSDF) ดังนั้นในกรณีที่ต้องการเพิ่มความเร็วในการทำงานโดยใช้แนวคิดของการคำนวณแบบท่อตรง (Pipeline) ไม่สามารถทำได้ เพราะทุกตัวดำเนินการต้องทำงานในทิศทางเดียวกัน สำหรับการทำการในลักษณะท่อตรงนั้น แต่ละตัวดำเนินการสามารถเริ่มทำงานได้โดยไม่ต้องรอให้การทำงานตัวดำเนินการก่อนหน้าเสร็จก่อน การคำนวณแบบเชื่อมตรงถูกออกแบบมาเพื่อใช้งานร่วมกับแนวคิดการทำท่อตรง แต่เนื่องจากการหารได้มีการพิสูจน์แล้วว่าไม่สามารถทำงานเริ่มต้นที่หลักที่มีนัยสำคัญต่ำสุดก่อนได้ ดังนั้น การคำนวณแบบเชื่อมตรงจึงเลือกใช้หลักที่มีนัยสำคัญสูงสุดทั้งหมด

ในการคำนวณทางคณิตศาสตร์แบบเชื่อมตรง ตัวถูกดำเนินการ (Operand) และทิศทางของผลลัพธ์ที่ได้จะเป็นไปตามลำดับทางคณิตศาสตร์ ผ่านทางหน่วยการคำนวณในลักษณะของตำแหน่งต่อตำแหน่ง (Digit-By-Digit) เริ่มจากตำแหน่งที่มีเลขนัยสำคัญสูงสุดก่อน ในระบบการคำนวณทางคณิตศาสตร์แบบเชื่อมตรงจะมีคุณลักษณะอย่างหนึ่งที่เรียกว่าค่าความหน่วงเชื่อมตรง (On-line Delay) แทนด้วยสัญลักณ์ δ ซึ่งจะเป็นตัวเลขจำนวนเต็มบวกขนาดเล็กเป็นค่าที่ระบุว่าผลลัพธ์จำนวน n ตำแหน่งแรกนั้นสามารถคำนวณออกมาได้จากตัวถูกดำเนินการ (n+δ) ตำแหน่งแรก ระบบจำนวนที่ใช้ในการแสดงผลจะเป็นระบบซ้ำซ้อน และระบบจำนวนแบบมีเครื่องหมายจึงได้ถูกนำมาใช้กับการคำนวณทางคณิตศาสตร์แบบเชื่อมตรง

1. การบวกแบบเชื่อมตรง

อัลกอริทึมข้างล่างนี้แสดงให้เห็นถึงลักษณะการทำงานของการบวกแบบเชื่อมตรง โดยกำหนดให้ β เป็นเลขฐานของระบบจำนวน และ D เป็นเซตจำกัดของตัวเลข ดังนี้

Algorithm

Input : $X := (X_a X_{a-1} \ldots)_\beta$ and $Y = (y_a y_{a-1} \ldots)_\beta$ where $x_i , y_i \in D$

Output : $Z := (z_a z_{a-1} \ldots)_\beta$ where $z_i \in D$

begin

$r_{a+1} := 0$;

$j := a$;

while $j \leq a$ do

$s_j := x_j + y_j$;

if $-2b \leq s_j < -b+1$ then $c_{j+1} := -1$; $r_j := s_j + \beta$;

endif;

if $-b+1 \leq s_j \leq b-1$ then $c_{j+1} := 0$; $r_j := s_j$;

endif;

if $b-1 < s_j \leq 2b$ then $c_{j+1} := 1$; $r_j := s_j - \beta$;

endif;

$z_{j+1} := c_{j+1} + r_{j+1}$;

$j := j - 1$;

enddo;

end;

ตัวอย่างเช่น กำหนดให้ใช้เลขฐานเท่ากับ $\beta = 5$ กับเซตของตัวเลข D = { -3,-2, -1, 0, 1, 2, 3} ในการทำการบวกค่าของ 708 กับ 766

		β^5	β^4	β^3	β^2	β^1	β^0
708	=		1	1	-2	1	3
766	=		1	1	1	-2	1
			2	2	-1	-1	4
			2	2	-1	-1	-1
Remainder		0	0	0	0	1	
Carry digit		0	2	2	-1	0	-1

$(0\ 2\ 2\ -1\ 0\ -1)_5 = (2*5^4) + (2*5^3) + (-1*5^2) + (-1)$

$= 1250 + 250 - 25 - 1 = 1474$

จากตัวอย่างข้างต้นผลลัพธ์ถูกสร้างขึ้นในลักษณะของตำแหน่งต่อตำแหน่ง เริ่มจากตำแหน่งที่มีเลขนัยสำคัญสูงสุดก่อน β^4 ด้วยค่าความหน่วงเชื่อมตรงเท่ากับหนึ่ง 1 จากตำแหน่ง β^4 ไม่สามารถที่จะสร้างผลลัพธ์ได้จนกว่าการกระจายของตัวทดของตำแหน่ง β^3 จะทราบค่า ในระบบจำนวนนี้จะเห็นได้ว่าผลกระทบของการกระจายของตัวทดจะถูกจำกัดให้มีแค่เพียงตัวเดียวที่ตำแหน่งที่มีเลขนัยสำคัญต่ำสุด

2. การลบแบบเชื่อมตรง

สำหรับการลบแบบเชื่อมตรงใช้แนวคิดเดียวกันกับการบวกแบบเชื่อมตรง เพียงแต่พิจารณาค่าลบของตัวถูกดำเนินการในระหว่างการทำงานเท่านั้น

3. การคูณแบบเชื่อมตรง

การคูณแบบเชื่อมตรงเป็นการใช้สองเทคนิครวมเข้าด้วยกัน คือ การเพิ่มค่าขึ้นสำหรับการคูณ (Incremental Multiplication) และระบบจำนวนซ้ำซ้อน ซึ่งการคูณแบบเชื่อมตรงค่าความหน่วงจะขึ้นอยู่กับขอบเขตของค่านำเข้า (input) ค่าความหน่วงนี้สามารถที่จะไม่นำมาพิจารณาได้ โดยเติมค่าศูนย์เท่ากับจำนวนค่าความหน่วงให้กับแต่ละตัวถูกดำเนินการทางด้านซ้าย นั่นคือทำให้แต่ละตัวถูกดำเนินการมีค่าน้อยกว่า $1/\beta^{\delta}$ ซึ่งจะเรียกว่า ขอบเขตของตัวถูกดำเนินการ (Operand Bound)

อัลกอริทึมข้างล่างนี้จะแสดงให้เห็นถึงลักษณะการทำงานของการคูณแบบเชื่อมตรง โดยกำหนดให้ β เป็นเลขฐานของระบบจำนวนและ D เป็นชุดตัวเลข มีค่าเท่ากับ {-b, -b+1, -b+2, …, 0, 1, …, b} ดังนี้

Algorithm M_R

Input: $A = (.a_{-1}a_{-2}\ldots)_\beta$ and $B := (.b_{-1}b_{-2}\ldots)_\beta$

Output: $X = (.x_{-1}x_{-2}\ldots)_\beta$ where $||X|| = \sum_{j \le -1} x_j\beta^j = ||A|| \cdot ||B||$

begin

$x_{-1} := x_{-2} := \ldots := x_{-\delta} := 0;$

$W_{-\delta} := 0;$

$j := -\delta-1;$

while $j \le -\delta-1$ do

$W_j := \beta(W_{j+1} - x_{j+1}) + A_jb_j + B_{j+1}a_j ;$

if $|W_j| \le b$ then $x_j := Sign(W_j) \lfloor |W_j| + 1/2 \rfloor$

else $x_j := Sign(W_j) \lfloor |W_j| \rfloor$ end if;

$j := j - 1;$

enddo;

end;

2.1.3 วิธีการเชิงเหตุการณ์ (Event Driven Approach)

การสร้างภาพมโนทัศน์ของอัลกอริทึมแสดงการทำงานมีหลายวิธี วิธีหนึ่งที่เหมาะสม คือ แสดงภาพมโนทัศน์ด้วยวิธีการเชิงเหตุการณ์ [3] วิธีนี้อาศัยเหตุการณ์ที่สนใจซึ่งเป็นจุดที่ทำการติดต่อกับส่วนประกอบอื่น โดยแบ่งเหตุการณ์ที่สนใจออกเป็น เหตุการณ์เข้า (Input Event) และเหตุการณ์ผลลัพธ์ (Output Event) เหตุการณ์เข้าเกี่ยวกับการขอข้อมูลเปรียบกับเป็นคำสั่งรับข้อมูลในการเขียนโปแกรม และเหตุการณ์ผลลัพธ์เปรียบกับคำสั่งในการแสดงผล ซึ่งเกี่ยวข้องกับสถานะของโปรแกรมหรือการแสดงผลลัพธ์ โดยเหตุการณ์ขาออกจะทำให้ส่วนการนำเสนอได้รับการแจ้งเตือนจากอัลกอริทึมให้ปรับปรุงภาพบนหน้าจอ เหตุการณ์ที่สนใจ เช่น การตอบสนองกับเหตุการณ์ที่เปลี่ยนแปลงภายในอัลกอริทึมแล้วแสดงออกมาเป็นกราฟิก วิธีการเชิงเหตุการณ์ประกอบด้วยสองรูปแบบ แบบที่หนึ่งคือ การให้ความสนใจต่อเหตุการณ์ที่เกิดขึ้น เช่น การสลับค่าของตัวแปรในอัลกอริทึมแบบเรียงลำดับ เป็นต้น และแบบที่สองความสนใจต่อเหตุการณ์ที่มีความสัมพันธ์กัน เช่น อัลกอริทึมการเรียงลำดับ จะเป็นการเรียงลำดับของความสูงของแท่งสี่เหลี่ยมที่ต่อเนื่องกันตามความสูง การแสดงภาพมโนทัศน์ของเหตุการณ์ที่มีการสลับค่าตัวแปรทำให้แท่งสี่เหลี่ยมที่ใช้ในการเปรียบเทียบเกิดการสลับค่าระหว่างกันเพื่อให้แสดงออกทางจอภาพตามเหตุการณ์นั้นได้

ตัวอย่างเช่น

```
int v[] = {3,5,2,9,6,4,1,8,0,7}, n=10, i, j;
void main(void) {
bsort.SendAlgoEvt("Input",n,v);
for (j=n; j>0; j- -)
        for (i=1; i<j; i++)
        if (v[i] > v[i+1]) {
        int temp = v[i]; v[i] = v[i-1]; v[i-1] = temp;
        bsort.SendAlgoEvt("Exchange",i,i-1);
        }
}
```

จากตัวอย่างมีสองเหตุการณ์ที่เกิดขึ้น เหตุการณ์แรก bsort.SendAlgoEvt ("Input",n,v) เป็นการเริ่มต้นการทำงานโดยการสร้างกราฟิกให้สัมพันธ์กับค่าตั้งต้นเหตุการณ์ที่สอง bsort.SendAlgoEvt("Exchange",i,i-1) เป็นเหตุการณ์สำหรับการสลับค่าของตัวแปรซึ่งทำให้เกิดการเปลี่ยนแปลงออกทางจอภาพ เป็นต้น

แนวคิดการเขียนโปรแกรมเชิงวัตถุ (Objected Oriented Programming, OOP) [4] [5] จะมองโปรแกรมเป็นรูปแบบของวัตถุที่มีพฤติกรรมเฉพาะของตัวเอง และมีความสัมพันธ์กับวัตถุอื่น การมองปัญหาในลักษณะนี้นำมาใช้แสดงภาพมโนทัศน์ด้วยวิธีการเชิงเหตุการณ์ได้เป็นอย่างดี กล่าวคือเมื่อมองตัวแปรที่สัมพันธ์กันเป็นวัตถุ (Object) การเปลี่ยนแปลงค่าของวัตถุจะทำให้มีผลกับค่าของวัตถุอื่นที่เกี่ยวข้องกันด้วย ซึ่งจะทำให้สามารถแสดงผลการทำงานภาพในอัลกอริทึมได้ชัดเจนมากยิ่งขึ้น

2.1.4 ทฤษฎีคอมไพเลอร์ (Compiler Theory) [6] [7] [8]

คอมไพเลอร์เป็นส่วนของการสร้างภาษาที่รับเอาภาษาโปรแกรมจากภาษาหนึ่งที่เขียนขึ้นโดยภาษาโปรแกรมระดับสูง (High Level Programming Language) เป็นข้อมูลนำเข้า (Input Data) หรือที่เรียกว่าภาษาต้นฉบับ (Source Language) และนำมาประมวลผลหรือที่เรียกว่าการคอมไพล์ (Compile) เพื่อให้ได้ผลลัพธ์ออกมาเป็นโปรแกรมที่เขียนด้วยอีกภาษาหนึ่งที่เรียกว่า ภาษาเป้าหมาย (Target Language) การทำงานของคอมไพเลอร์ ประกอบด้วยโครงสร้างการทำงานดังต่อไปนี้

1. การวิเคราะห์คำ (Lexical Analyzer)

หน้าที่ของส่วนการวิเคราะห์คำ คือ อ่านโปรแกรมต้นฉบับ (Source Program) ทีละตัวอักษร และแบ่งแถวหรือลำดับของตัวอักษรที่ได้รับออกเป็นคำ เรียกคำเหล่านั้นว่า โทเค็น (Token)

2. การวิเคราะห์รูปประโยค (Syntax Analyzer)

เป็นส่วนของการตรวจสอบว่า Token แต่ละตัวในโปรแกรมต้นฉบับ นั้นถูกจัดเรียงไว้ถูกต้องตามหลักการที่กำหนดไว้โดยไวยากรณ์ (Grammar) ของภาษาที่กำหนดหรือไม่

3. การส่วนวิเคราะห์ความหมาย (Semantic Analyzer)

โดยหน้าที่สำคัญของ ส่วนการวิเคราะห์ความหมาย คือ การตรวจสอบเพื่อดูชนิด (Type) ของตัวแปรที่นำมาใช้นั้นถูกต้องหรือไม่

4. การสร้างโค้ดภาษากลาง (Intermediate Code Generator)

เป็นส่วนของการแปลงโปรแกรมต้นฉบับให้เป็นรูปแบบของโค้ดภาษากลาง

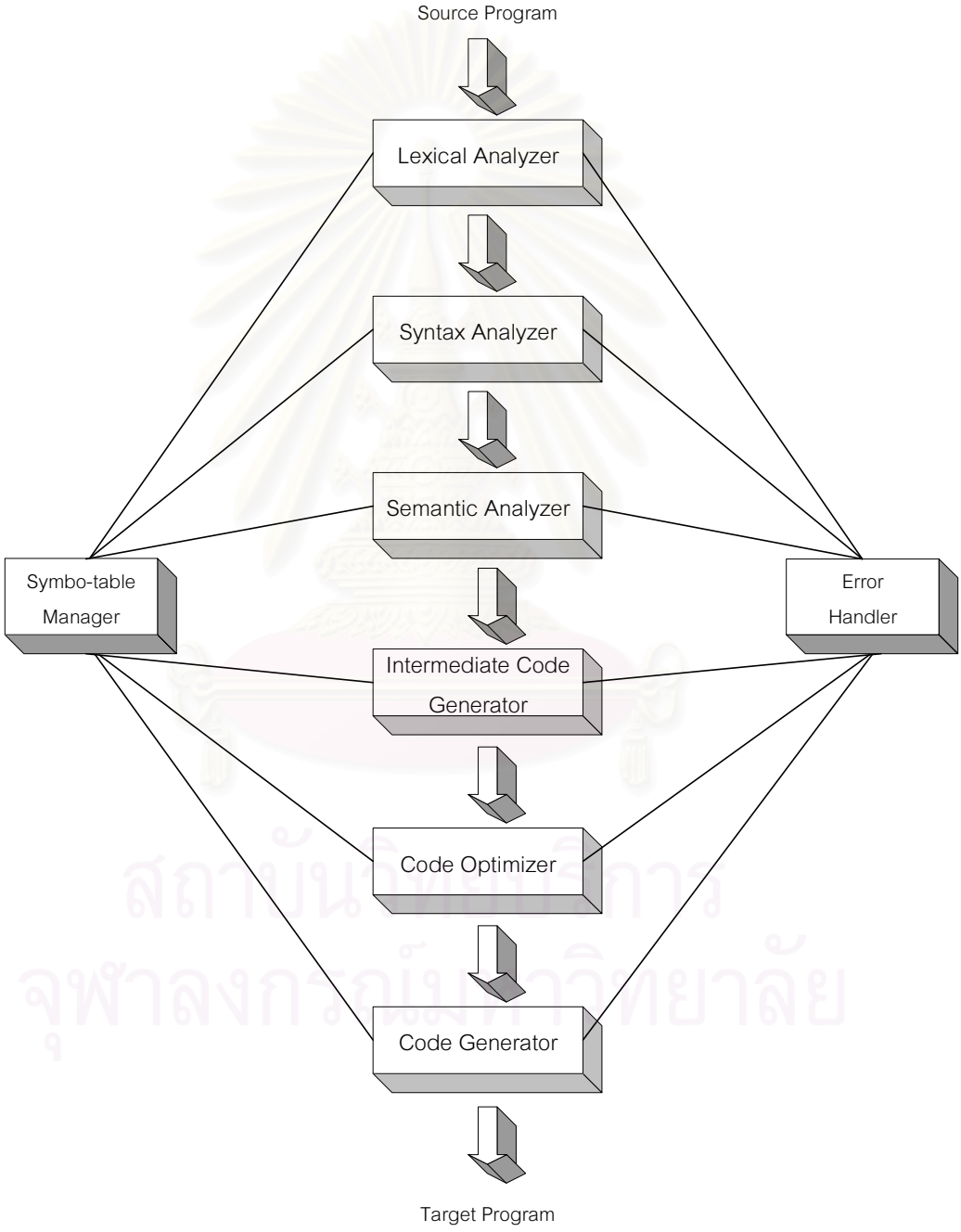
5. การสร้างโค้ดให้มีประสิทธิภาพ (Code Optimizer)

เป็นการใช้เทคนิคพิเศษเพื่อปรับปรุงให้โปรแกรมเป้าหมายนั้นให้สามารถทำงานได้อย่างมีปะสิทธิภาพมากยิ่งขึ้น ไม่มีการทำงานหรือคำนวณใดๆที่ไม่จำเป็น ซึ่งจะทำให้สามารถประมวลผลได้เร็วขึ้น เป็นต้น

6. การสร้างโค้ดเป้าหมาย (Code Generator)

หน้าที่ของการสร้างโค้ดเป้าหมาย คือการสร้าง อ็อบเจกต์โค้ด (Object Code) จากโค้ดภาษากลาง ซึ่งอ็อบเจกต์โค้ด อาจอยู่ในรูปแบบของภาษาโปรแกรมระดับล่าง หรือภาษาเครื่อง (Machine Language)

การทำงานแต่ละขั้นตอนมีความสัมพันธ์ที่เกี่ยวข้องกันสามารถแสดงขั้นตอนการทำงานของคอมไพเลอร์โดยสรุปได้ดังรูปที่ 2.1



รูปที่ 2.1 ขั้นตอนการทำงานของคอมไพเลอร์

2.2 งานวิจัยที่เกี่ยวข้อง

งานวิจัยสำหรับแสดงภาพมโนทัศน์อัลกอริทึม ซึ่งแสดงภาพเคลื่อนไหว และแสดงการทำงานของอัลกอริทึม ทำให้ผู้ศึกษาเข้าใจได้ง่าย และเห็นพฤติกรรมการทำงานโดยทั่วไปสามารถแบ่งออกได้เป็นสองลักษณะคือ ระบบแสดงภาพมโนทัศน์ที่แยกออกเป็นระบบย่อยๆ เพื่อความสะดวกในการพัฒนา และระบบแสดงภาพมโนทัศน์อัลกอริทึมที่เน้นพัฒนาในส่วนการแสดงผลของการสร้างภาพมโนทัศน์อัลกอริทึม เป็นต้น

2.2.1 บัลซาร์-วัน และ บัลซาร์-ทู (BALSA-I and BALSA-II) [9] [10]

บัลซาร์-วัน เป็นโปรแกรมที่ถูกพัฒนาให้ใช้งานบนเครื่องคอมพิวเตอร์อพอลโล (Apollo Workstation) และนำมาใช้ในการเรียนการสอนในสาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ เช่น การเขียนโปรแกรมโครงสร้างข้อมูลกราฟิก และงานวิจัยด้านการวิเคราะห์และออกแบบอัลกอริทึม ของมหาวิทยาลัยบราวน์ตั้งแต่ปีค.ศ. 1983 บัลซาร์-ทู เป็นซอฟต์แวร์สร้างภาพมโนทัศน์อัลกอริทึมที่พัฒนาต่อจาก บัลซาร์-วัน ทำงานบนเครื่องคอมพิวเตอร์แมคอินทอช โดยผู้ใช้งานสามารถดูการทำงานของอัลกอริทึมผ่านการแสดงผลได้หลากหลายรูปแบบ

2.2.2 การออกแบบและพัฒนาระบบจินตทัศน์อัลกอริทึมสำหรับปัญหาทางทฤษฎีกราฟ [11]

งานวิจัยนี้เพื่อพัฒนาระบบจินตทัศน์อัลกอริทึมโดยครอบคลุมอัลกอริทึมสำหรับซนิดไม่มีทิศทาง เช่น อัลกอริทึมการค้นหาแนวลึก การค้นหาในแนวกว้าง การหาเส้นทางสั้นที่สุดของดิสตรา เป็นต้น โดยเสนอการจินตทัศน์ในสี่มุมมองคือ มุมมองแสดงตัวกราฟ และการเปลี่ยนแปลงสถานะของเส้นเชื่อม มุมมองแสดงความคืบหน้าของการทำงาน มุมมองแสดงจำนวนเส้นเชื่อมที่ต้องพิจารณาระหว่างการทำงาน และมุมมองแสดงความลึกของจุดระหว่างแวะผ่านจุดในกราฟ

2.2.3 ระบบจินตทัศน์อัลกอริทึมของปัญหาทางด้านเรขาคณิตเชิงคำนวณ [12]

งานวิจัยนี้เพื่อพัฒนาระบบจินตทัศน์อัลกอริทึมสำหรับปัญหาเรขาคณิตเชิงคำนวณในสองมิติ ระบบนี้มีส่วนอัลกอริทึมหกส่วนเพื่อการจินตทัศน์ปัญหาเปลือกนูน ได้แก่ อัลกอริทึมแบบห่อของขวัญของ เจวิส (Javis) อัลกอริทึมแบบกราดตรวจของ การ์แฮม อัลกอริทึมแบบค่อยๆเพิ่มจุด อัลกอริทึมแบบแบ่งแยกแล้วเอาชนะ อัลกอริทึมการหาเปลือกนูนแบบเร็ว อัลกอริทึมแบบจำกัดและปัญหาการค้นหาในพิสัย

2.2.4 ระบบจินตทัศน์อัลกอริทึมการเรียงลำดับข้อมูล [13]

งานวิจัยนี้เพื่อพัฒนาระบบจินตทัศน์อัลกอริทึมโดยใช้ภาพ และการเปลี่ยนแปลงของภาพเป็นสื่อแสดงขั้นตอนการทำงานอัลกอริทึมการเรียงลำดับ 7 วิธี คือ การเรียงลำดับข้อมูลแบบเลือก แบบแทรก แบบฟอง แบบเชลล์ แบบเร็ว แบบฮีบ และ แบบผสาน ประกอบมุมมองการนำเสนอ 3 รูปแบบคือ มุมมองแบบจุด แบบแท่ง และแบบแถบสี

2.2.5 เครื่องมือ "Jeliot 3" [14]

Jeliot 3 เป็นเครื่องมือที่จัดอยู่ในกลุ่มเครื่องมือสร้างภาพมโนทัศน์สำหรับช่วยในการศึกษาด้านการเขียนโปรแกรมพัฒนาด้วยโปรแกรมภาษาจาวา เพื่อช่วยเพิ่มประสิทธิภาพในด้านการเรียนการสอน ทางด้านการเขียนโปรแกรมเบื้องต้นด้วยภาษาจาวา ทำให้นักศึกษาสามารถตรวจสอบการทำงานได้อย่างเป็นขั้นตอน ซึ่งใช้ในการอธิบายแนวคิดและโครงสร้างของการเขียนโปรแกรม เครื่องมือจะทำการสร้างชุดคำสั่งอินเตอร์มิเดียทที่เรียกว่า MCode จากตัวโปรแกรมต้นฉบับสำหรับใช้ตรวจสอบสถานะการทำงานในขณะรันไทม์ (Runtime) โดยที่ MCode จะถูกนำไปวาดภาพการทำงานของโปรแกรมต้นฉบับผ่านตัวแสดงผลกราฟิก ซึ่งการเขียนโปรแกรมผ่านเครื่องมือยังสามารถใช้แนวคิดการเขียนโปรแกรมเชิงวัตถุ เช่น การสืบทอดคุณสมบัติ (Inheritance) เป็นต้น นำมาแสดงภาพเคลื่อนไหวโดยแสดงขั้นตอนการทำงานผ่านส่วนประสานงานกับผู้ใช้ ซึ่งเป็นเครื่องมือที่เหมาะกับการศึกษาการเขียนโปรแกรมเบื้องต้นด้วยโปรแกรมภาษาจาวาได้เป็นอย่างดี แต่ไม่สามารถทำงานเฉพาะอย่างที่ต้องการเห็นการเปลี่ยนแปลงสถานะของข้อมูลเข้าและออกจากตัวดำเนินการโดยแสดงในลักษณะการสร้างเส้นความสัมพันธ์ของตัวแปรและคำสั่งที่เกี่ยวข้องได้

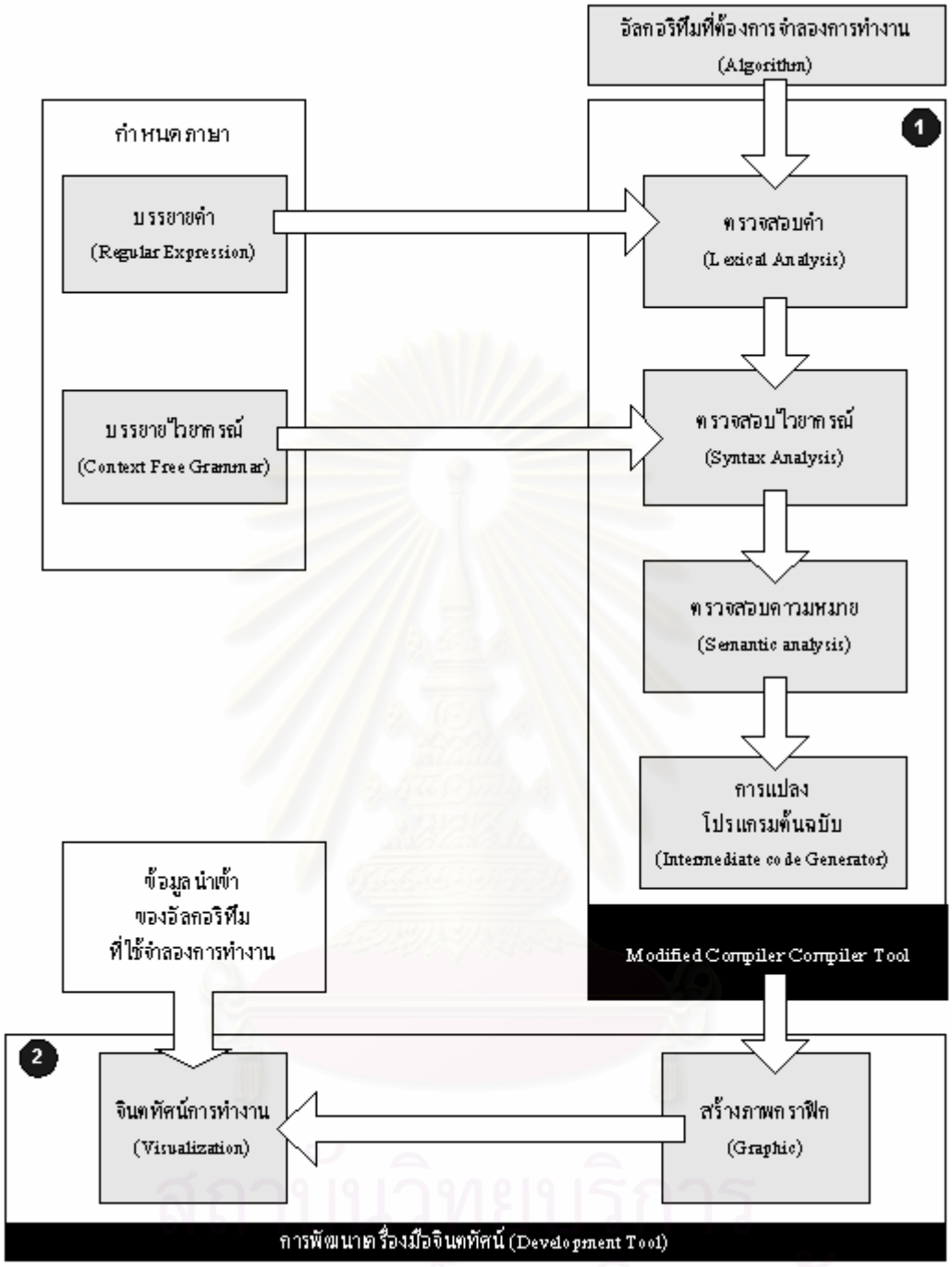
บทที่ 3

แนวคิดและวิธีการวิจัย

วิทยานิพนธ์ฉบับนี้ได้ออกแบบและพัฒนาเครื่องมือสร้างภาพมโนทัศน์เพื่อสนับสนุนการออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิต โดยการจำลองอัลกอริทึมที่เขียนอยู่ในรูปแบบโปรแกรมการทำงานแสดงออกเป็นรูปแผนภาพเคลื่อนไหว ผู้ออกแบบสามารถเรียนรู้การทำงานของอัลกอริทึมได้จากรูปแบบประโยคคำสั่งมากกว่าที่จะเป็นเฉพาะการใช้ค่าของข้อมูลนำเข้าเพียงอย่างเดียว ทำให้เห็นการเปลี่ยนแปลงค่าระหว่างสถานะการทำงานที่กำลังศึกษาเป็นไปอย่างมีลำดับขั้นตอน และถูกต้องชัดเจน ลดเวลาในการทำความเข้าใจ ตลอดจนใช้วิธีปรับเปลี่ยนวิธีการแก้ปัญหาให้ตรงกับความต้องการได้อย่างเหมาะสม

ผู้วิจัยมีแนวคิดในการออกแบบและพัฒนาเครื่องมือที่สามารถแปลงอัลกอริทึม ที่เขียนอยู่ในลักษณะโปรแกรมการทำงานตามลำดับของคำสั่งหรือลำดับของการคำนวณ เพื่อแสดงให้เห็นการส่งค่าในระดับดิจิตได้ชัดเจน ก่อนนำอัลกอริทึมไปสร้างวงจรการคำนวณ โดยสร้างเป็นภาพมโนทัศน์แสดงถึงการไหลของข้อมูลระดับดิจิตที่ผ่านไปตามคำสั่งต่างๆ ที่เกี่ยวข้องกับตัวแปรนำเข้าและตัวแปรผลลัพธ์ได้ โดยแนวทางการวิจัยสามารถสรุปแต่ละขั้นตอน รายละเอียดแสดงดังต่อไปนี้

1. กำหนดภาษาคำสั่งพื้นฐานสำหรับการบรรยายการทำงานของอัลกอริทึมแบบง่าย
2. บรรยายคำทั้งหมดในรูปแบบของคำนิยามในภาษา
3. บรรยายไวยากรณ์ของภาษา
4. สร้างโปรแกรมแปลภาษาต้นฉบับให้อยู่ในรูปแบบชุดคำสั่งอินเตอร์มิเดียท (Intermediate Instruction)
5. การสร้างเครื่องมือ
   5.1 การออกแบบเครื่องมือ
   5.2 แปลงคำสั่งที่ได้ให้อยู่ในรูปแบบกราฟิก
   5.3 สร้างภาพมโนทัศน์การทำงานของอัลกอริทึม



รูปที่ 3.1 แสดงแผนภาพขั้นตอนการทำงานโดยรวมของเครื่องมือ

3.1 กำหนดภาษาคำสั่งพื้นฐาน

ในการสร้างคำสั่งพื้นฐานของภาษานี้ จะออกแบบคำสั่งให้ครอบคลุมการทำงานโดยทั่วไปของอัลกอริทึมการคำนวณแบบเชื่อมตรงระดับดิจิต โดยสามารถแบ่งประเภทของคำสั่งออกได้ตามลักษณะการทำงานดังนี้

คำสั่งประกาศ (Declaration Instruction)

เป็นกลุ่มของคำสั่งที่ใช้ในการประกาศ และตั้งชื่อ การประกาศในที่นี้รวมถึงการประกาศชื่อฟังก์ชันการทำงานภายในอัลกอริทึม และชื่อของตัวแปรพร้อมระบุประเภทของตัวแปร โดยในที่นี้ ตัวแปรที่ใช้ในงานวิจัยนี้เป็นตัวแปรระดับดิจิตที่สอดคล้องกับทฤษฎีการคำนวนแบบเชื่อมตรง และครอบคลุมกรณีที่ตัวแปรเป็นแบบแถวลำดับ (Array) ได้ด้วย ตัวแปรที่กำหนดเป็นตัวแปรนำเข้า และตัวแปรผลลัพธ์ต้องมีการระบุอย่างชัดเจนในอัลกอริทึมด้วยเช่นเดียวกัน

คำสั่งกำหนดค่า (Assignment Instruction)

เป็นกลุ่มของคำสั่งที่ใช้ในการกำหนดค่าให้กับตัวแปรต่างๆ ในอัลกอริทึม การกำหนดค่านี้ สามารถกำหนดค่าได้โดยใช้การบรรยายการคำนวณ การกำหนดค่าจากผลลัพธ์ของอัลกอริทึมอื่น การกำหนดค่าการเลื่อนตำแหน่งของตัวชี้ (Pointer) ด้วย และรวมถึงการกำหนดค่าจากการนำเข้าในกรณีที่ตัวแปรนั้นเป็นตัวแปรนำเข้า

คำสั่งควบคุมเส้นทางการทำงาน (Control Flow Instruction)

เป็นกลุ่มของคำสั่งที่กำหนดเส้นทางการทำงาน โดยปกติคำสั่งทั้งหมดจะถูกทำงานแบบลำดับ ทีละคำสั่ง คำสั่งควบคุมการทำงานนี้ จะประกอบด้วย คำสั่งการทำงานตามข้อกำหนด (Condition Instruction) คำสั่งการทำงานแบบวนซ้ำ (While-loop Instruction) คำสั่งเคลื่อนย้ายจุดทำงาน (Jump Instruction) เป็นต้น

3.2 การบรรยายคำ

การบรรยายคำในภาษาที่กำหนด ในส่วนการบรรยายจะใช้คำบรรยายแบบนิยามสม่ำเสมอ อ้างอิงตามอักขระ (Alphabet) ของภาษา โดยการบรรยายแบบนี้จะบรรยายด้วยกฎ

Word $\longrightarrow$ regular expression on variables and/or characters in alphabet

โดยคำในภาษามีความหมายรวมถึง คำที่ถูกจอง (Reserved Word) ชื่อตัวแปรในภาษา (Variable Name or Identifier) รูปแบบค่าคงที่ของชนิดข้อมูลต่างๆ และยังรวมถึงคำบรรยายการคำนวณในรูปแบบของตรรกะด้วย การบรรยายคำจะกำหนดในแฟ้มข้อมูลเฉพาะของเครื่องมือช่วยสร้างตัวแปลภาษา รายละเอียดแสดงดังภาคผนวก ก.

3.3 การบรรยายไวยากรณ์

การบรรยายไวยากรณ์จะบรรยายด้วย ไวยากรณ์ที่ไม่ขึ้นกับบริบท (Context-Free Grammar) โดยสิ่งที่ต้องทำการบรรยายในที่นี้คือ ข้อกำหนดต่างๆ ของการรวมกันของคำในคำสั่ง ลักษณะของการบรรยายด้วยไวยากรณ์ที่ไม่ขึ้นกับบริบทนั้นใช้การบรรยายด้วยกฎ

Instruction → sequence of variables and/or words

เช่นในการบรรยาย คำสั่งกำหนดค่าให้กับตัวแปร สามารถบรรยายได้ดังนี้

Assignment → Identifier Equal Expression

โดยในการบรรยายคำจะต้องมีการบรรยายไว้แล้วว่า

Character → A | B | C | ... | Z | a | b | c | ... | z

Digit → 0 | 1 | 2 | 3 | 4 | ... | 9

Identifier → (Character)(Character | Digit)*

Equal → =

Expression → Identifier Operator Expression | Identifier

Operator → and | or | xor | not เป็นต้น

การกำหนดรูปแบบข้อมูลของเอกสารข้อความใดๆ ที่มีรูปแบบเขตข้อมูลที่ชัดเจน มีลักษณะที่เป็นการถูกกำหนดได้จากตัวอักษรที่ปรากฏอยู่ในข้อมูลนั้นอยู่แล้ว ซึ่งสามารถใช้เป็นตัวอักษรใดก็ได้แต่ปกตินิยมใช้ตัวอักษรจุลภาค (Comma) มากำหนดวิธีการกำหนดรูปแบบนี้จะอาศัยตัวอักษรดังกล่าวมากำหนดเขตข้อมูลแต่ละช่วง การบรรยายไวยากรณ์จะกำหนดในแฟ้มข้อมูลเฉพาะของเครื่องมือช่วยสร้างตัวแปลภาษา รายละเอียดแสดงดังภาคผนวก ก.

3.4 การสร้างโปรแกรมแปลภาษา

ในขั้นตอนนี้เป็นการสร้างโปรแกรมแปลภาษาเพื่อใช้ในการตีความภาษาที่เขียนไว้ และตรวจสอบว่าเขียนโปรแกรมได้ถูกต้องตามไวยากรณ์หรือไม่ ถ้าถูกต้องโปรแกรมจะทำการสร้างโครงสร้างที่ง่ายต่อความเข้าใจการทำงานของอัลกอริทึม โดยในที่นี้ภาษาที่เลือกมาแสดงคือ คำสั่งอินเตอร์มิเดียท (Intermediate Instruction) โดยแต่ละคำสั่งจะเกี่ยวข้องกับตัวแปรไม่เกินสามตัวแปรเท่านั้น โดยในสี่ขั้นตอนที่กล่าวมานี้จะเลือกใช้โปรแกรมสำเร็จรูป และปรับปรุงให้สอดคล้องกับการทำงาน

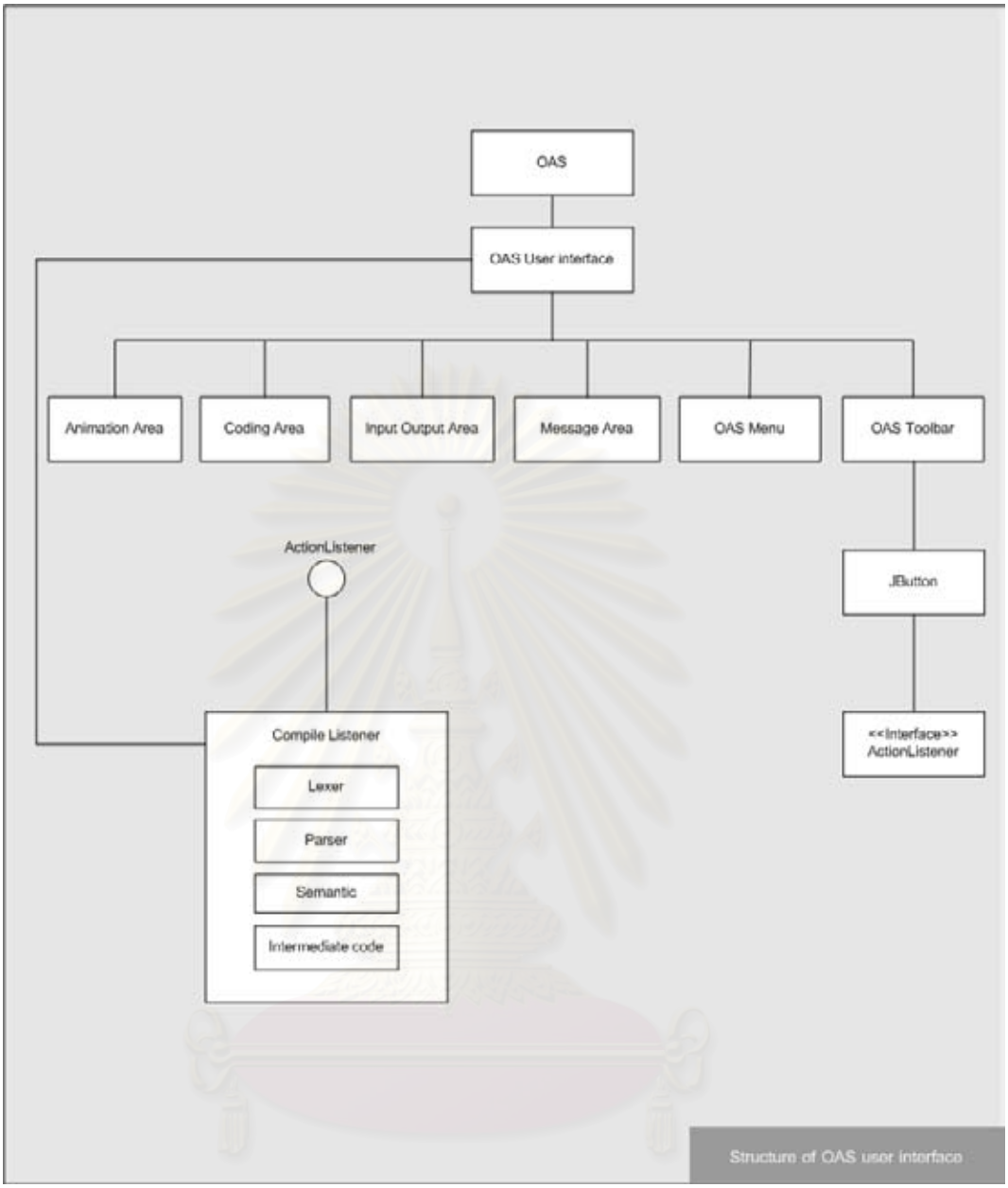
3.5 การสร้างเครื่องมือ

การออกแบบและพัฒนาสร้างเครื่องมือใช้วิธีการเชิงเหตุการณ์ในการออกแบบ โดยเลือกใช้การเขียนโปรแกรมเชิงวัตถุ โดยสามารถแบ่งขั้นตอนของการสร้างเครื่องมือออกเป็นสามส่วนได้ดังนี้

3.5.1 การออกแบบเครื่องมือ

ส่วนควบคุมและส่วนประสานงานกับผู้ใช้ (Control and Graphic User Interface) เพื่อให้เกิดประสิทธิภาพมากที่สุดในนำชุดคำสั่งสร้างภาพมโนทัศน์ของอัลกอริทึม เพื่อให้เกิดประโยนช์ทั้งผู้สอน และผู้ศึกษา

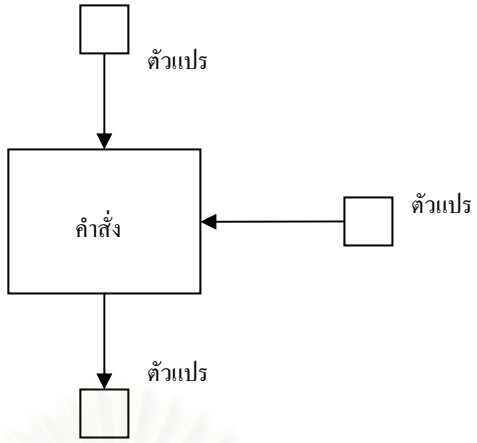
ผู้จัดทำวิทยานิพนธ์มีแนวคิดในการประมวลผลการแสดงภาพมโนทัศน์ของอัลกอริทึมด้วยการรับคำสั่งอัลกอริทึมที่อยู่ในรูปแบบโปรแกรมการทำงานของภาษาที่พัฒนาขึ้น จากนั้นทำการสร้างโครงสร้างความสัมพันธ์ของการทำงานระหว่างตัวแปรนำเข้า ตัวแปรผลลัพธ์ และตัวแปรที่เกี่ยวข้องกับตัวแปรนำเข้า และตัวแปรผลลัพธ์ ตลอดจนฟังก์ชันการทำงานที่ใช้แทนหน่วยย่อยของการคำนวณต่างๆ นำมาแสดงผลในรูปแบบของกราฟิกผ่านทางส่วนต่อประสาน (User Interface) ประกอบไปด้วยส่วนต่างๆ ดังนี้ ส่วนของเมนู (OAS Menu) เช่น สร้างแฟ้มข้อมูลโปรแกรมต้นฉบับใหม่ เปิดแฟ้มข้อมูล บันทึกแฟ้มข้อมูล เป็นต้น ส่วนทูลบาร์ (OAS Toolbar) คือส่วนการรับคำสั่งเพื่อใช้ควบคุมการแสดงภาพมโนทัศน์ เช่น การแปลโปรแกรมต้นฉบับซึ่งภายในประกอบด้วยขั้นตอนการทำงานของขบวนการแปลภาษา การสั่งแสดงภาพมโนทัศน์ การสั่งหยุดแสดงภาพมโนทัศน์ เป็นต้น ส่วนการรับข้อมูลตัวแปรนำเข้า และตัวแปรผลลัพธ์ (Input and Output Area) ซึ่งใช้สำหรับคำนวณค่าที่ผู้ใช้ป้อนข้อมูลนำเข้า และส่วนแสดงผลลัพธ์ของคำตอบที่ได้จากส่วนแสดงภาพมโนทัศน์การทำงานของอัลกอริทึม และส่วนแสดงผลการทำงาน (Message Area) เช่น รายงานสถานะของการใช้โปรแกรมว่าถูกต้องหรือไม่ เป็นต้น โครงสร้างส่วนประสานงานกับผู้ใช้ แสดงดังรูปที่ 3.2



รูปที่ 3.2 โครงสร้างส่วนประสานงานกับผู้ใช้

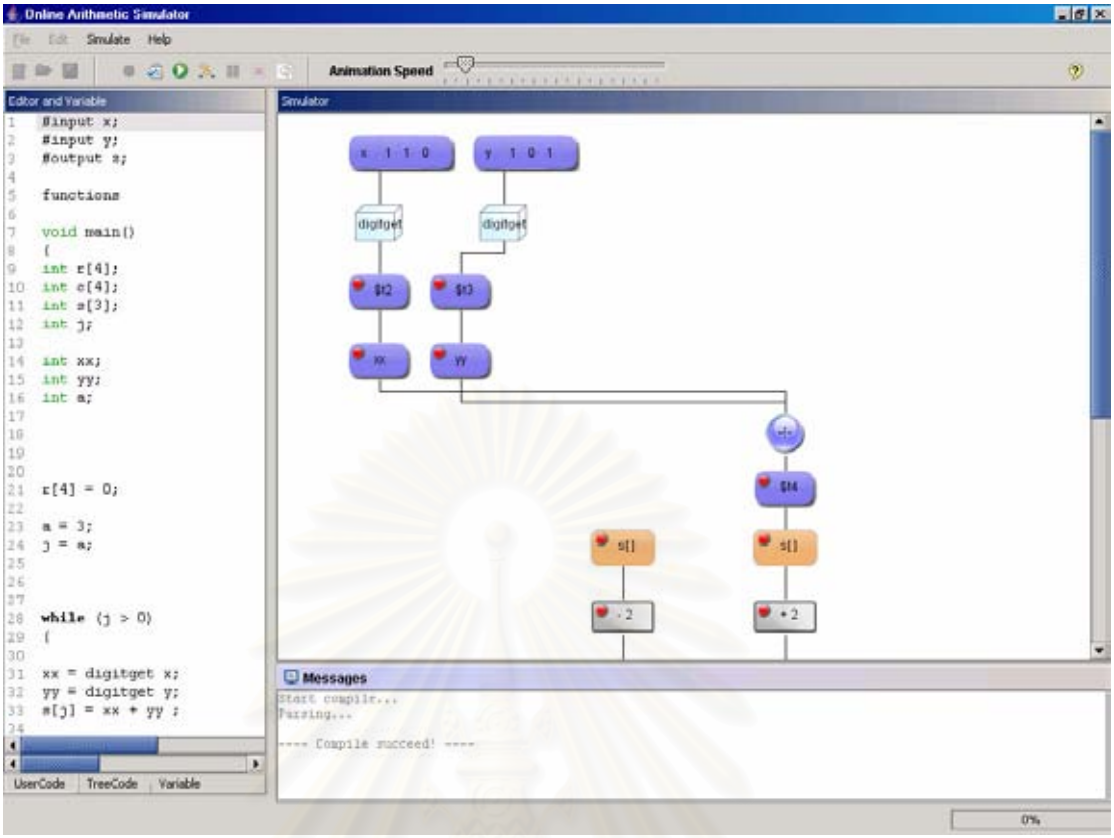
3.5.2 การแปลงคำสั่งให้เป็นกราฟิก (Converter)

ในขั้นตอนนี้จะรับชุดคำสั่งอินเตอร์มิเดียทที่ผ่านขั้นตอนการตรวจสอบด้วยโปรแกรมแปลภาษา จากนั้นทำการสร้างภาพแทนคำสั่ง โดยภาพที่เกิดขึ้นหนึ่งคำสั่งจะสอดคล้องกับตัวแปร โดยแต่ละคำสั่งจะถูกกำหนดมาตามลำดับที่ได้จากลำดับชุดคำสั่งอินเตอร์มิเดียท ดังรูปที่ 3.3



รูปที่ 3.3 การสร้างภาพจากคำสั่ง

จากนั้นจะนำไปสร้างโครงสร้างความสัมพันธ์ของตัวแปร ดังรูปที่ 3.4 ที่สอดคล้องกับคำสั่งโปรแกรมการทำงานที่รับเข้ามา เพื่อนำไปแสดงภาพมโนทัศน์ผ่านส่วนประสานงานกับผู้ใช้ การสร้างภาพจากคำสั่งเป็นส่วนการทำงานที่อยู่ระหว่างกลางของขั้นตอนการแปลงชุดคำสั่งอินเตอร์มิเดียทเพื่อสร้างโครงสร้างความสัมพันธ์ของตัวแปรกับขั้นตอนการทำงานของส่วนแสดงภาพมโนทัศน์ การทำงานเพื่อแปลงคำสั่งแสดงผลการทำงานเป็นขั้นตอนที่ทำให้อยู่ในรูปแบบที่ส่วนประมวลผลภาพมโนทัศน์เข้าใจ เพื่อนำไปสร้างโครงสร้างความสัมพันธ์และชุดคำสั่งสำหรับแสดงภาพมโนทัศน์ จากนั้นจึงประมวลผลตามคำสั่งต่อไป



รูปที่ 3.4 ภาพความสัมพันธ์ของตัวแปรและคำสั่ง

ภาพมโนทัศน์ (Visualization) ในขั้นตอนนี้เป็นการแสดงการทำงานของอัลกอริทึมโดยใช้ส่วนกราฟิกที่ได้รับการแปลงมาจากอัลกอริทึมเรียบร้อยแล้ว แสดงขั้นตอนการทำงานโดยใช้สี, รูปภาพ และการเปลี่ยนแปลงของค่าตัวแปรมาช่วยในการนำเสนอเพื่อให้ผู้ใช้งานสามารถทำความเข้าใจ และติดตามพฤติกรรมหรือสถานะการทำงานของอัลกอริทึมได้ชัดเจนมากยิ่งขึ้น การทำงานทั้งหมดของโปรแกรมที่พัฒนาจะทำงานในกราฟิกโหมด ในการแสดงภาพมโนทัศน์การทำงานของอัลกอริทึมที่ออกแบบนั้น ผู้ใช้สามารถสั่งให้โปรแกรมทำงานทีละขั้นตอนเพื่อดูผลลัพธ์ที่เกิดขึ้นในแต่ละช่วงเวลา หรือทำงานแบบต่อเนื่องทุกขั้นตอนจนแล้วเสร็จ โดยผู้ใช้เป็นผู้ควบคุมการทำงานผ่านทางส่วนต่อประสานผู้ใช้ การทำงานทั้งหมดของโปรแกรมที่พัฒนาจะทำงานในกราฟิกโหมด

ตัวอย่าง อัลกอริทึมการบวกแบบเชื่อมตรงโดยใช้ตัวแปรนำเข้าสองตัวแปร มีขั้นตอนการทำงานเป็นลำดับ ดังนี้

รูปแบบ Algorithm การบวกแบบเชื่อมตรงตัวแปร x และ y

ให้ INPUT $X = (x_1x_2x_3 \dots x_n)$ Digit

ให้ INPUT $Y = (y_1y_2y_3 \dots y_n)$ Digit

ให้ OUTPUT $Z = (z_1z_2z_3 \dots z_{2n})$ Digit

โดยที่ $z_{i+1} := c_{i+1} + r_{i+1}$;

ตัวอย่างเช่น X = 110, Y = 101 ผลลัพธ์คือ Z = 110 -1

```
input redundant x;
input redundant y;
output z;
globalvariables
void main()
{
int r[4]; int c[3];  int s[3];int j;int xx;int yy;int a;
r[4] = 0;
j = 3;
a = j ;
r[j] = 0;
while (j<=a)
{
xx = digitget x;  yy = digitget y;  s[j] = xx + yy ;
if (-2*2 <= s[j] and s[j] < -2 + 1 )
{ c[j+1] = -1;
  r[j] = s[j] + 2 ;
}
if (-2+1 <= s[j] and s[j] < 2 - 1 )
{ c[j+1] = 0;
  r[j] = s[j]  ;
}
if (2-1 < s[j] and s[j] < 2 * 2 )
{ c[j+1] = 1;
  r[j] = s[j] - 2  ;
}
z = c[j+1] + r[j+1] ;   } }
```

รูปที่ 3.5 โปรแกรมต้นฉบับเขียนด้วยอัลกอริทึมการบวกแบบเชื่อมตรงสองตัวแปร

1. การรับคำสั่งอัลกอริทึม

เป็นขั้นตอนแรกของการทำงานโดยเขียนโปรแกรมให้อยู่ในรูปแบบของภาษาที่ได้นิยามไว้แล้ว ดังรูปที่ 3.5

2. ขั้นตอนตรวจสอบคำ

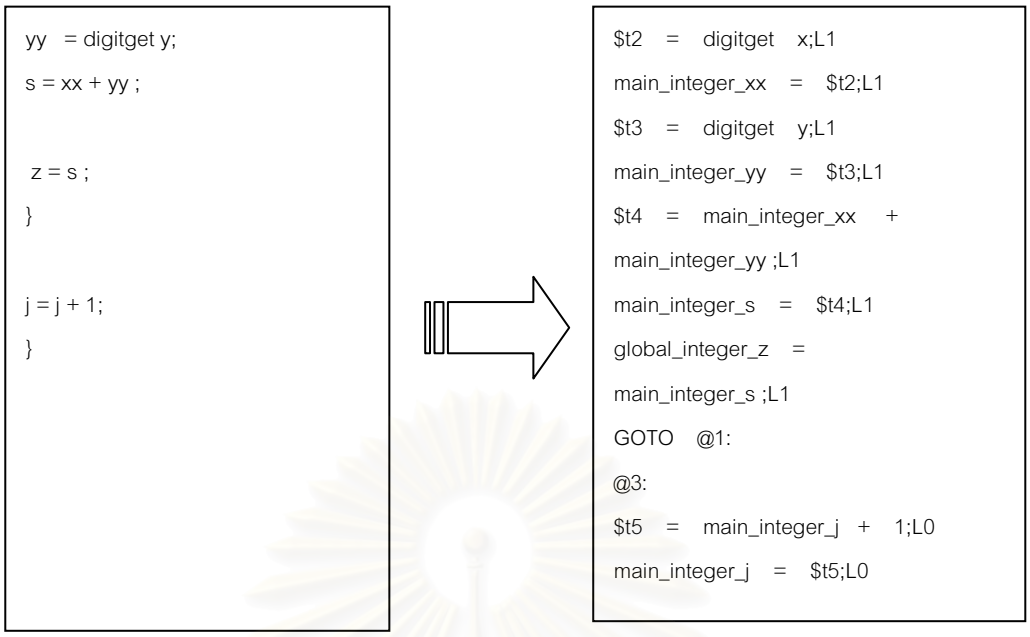
เป็นขั้นตอนในการบรรยาย และตรวจสอบคำในคำสั่งอัลกอริทึมว่าถูกต้องตามที่ได้นิยามไว้หรือไม่ โดยใช้ Regular expression ในการบรรยายคำ เช่น อักษรตัวหนาในขั้นตอนป้อนคำสั่งอัลกอริทึมการบวกแบบเชื่อมตรงสองตัวแปรเป็นคำสงวน ไม่สามารถใช้ในการตั้งชื่อตัวแปรได้ เป็นต้น

3. ขั้นตอนตรวจสอบไวยากรณ์

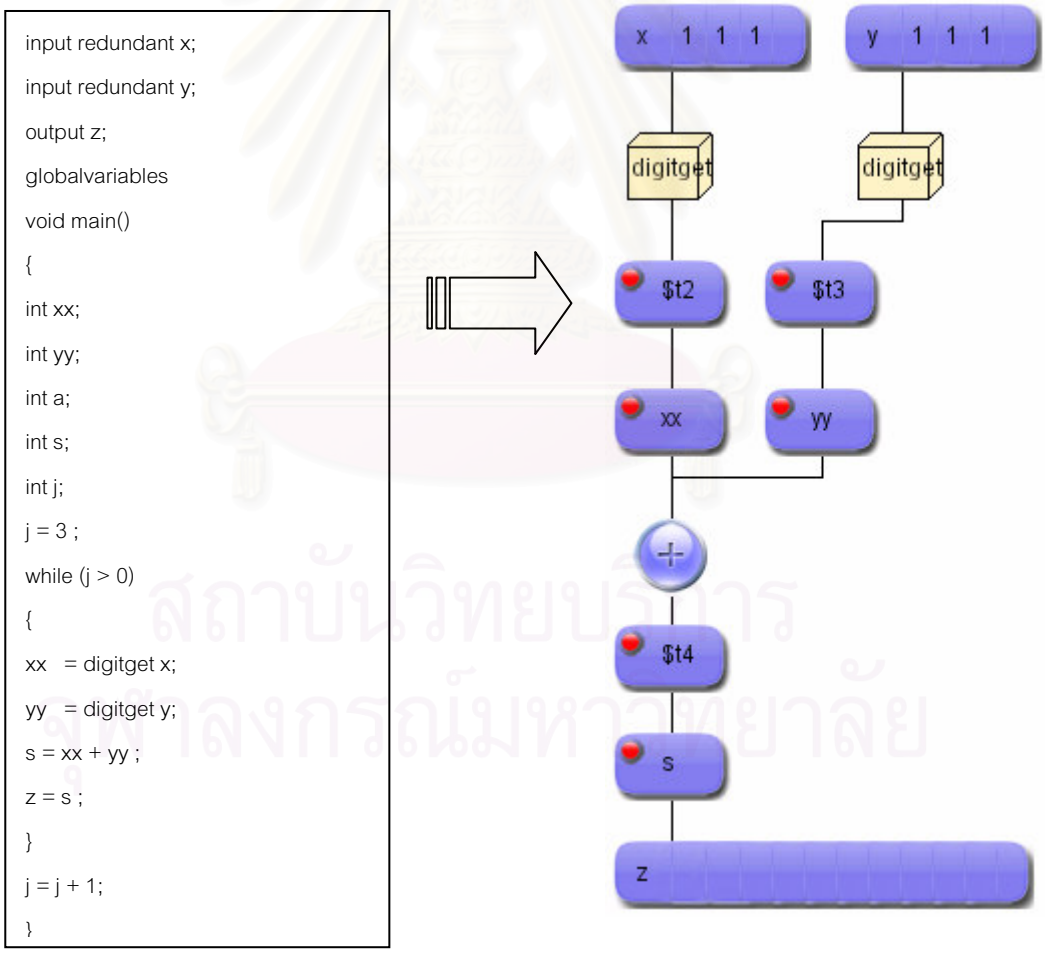
เป็นขั้นตอนถัดมาจากขั้นตอนการตรวจสอบคำ เพื่อตรวจสอบประโยคหรือคำสั่ง โดยใช้ Context Free Grammar ว่าประโยคที่เขียนนั้นถูกต้องตามหลักไวยากรณ์หรือไม่ หากรูปประโยคที่เขียนไม่ถูกต้องจะฟ้องว่ามีข้อผิดพลาดทางไวยากรณ์ให้ผู้ใช้งานทราบ

4. ขั้นตอนสร้างโปรแกรมแปลภาษา

เป็นขั้นตอนที่ทำหลังจากการตรวจสอบคำ และตรวจสอบไวยากรณ์ จากนั้นสร้างโปรแกรมแปลภาษาเพื่อใช้ในการตีความภาษาที่เขียนแปลงเป็นคำสั่งอินเตอร์มิเดียท ตามตัวอย่าง ดังรูปที่ 3.6



รูปที่ 3.6 โปรแกรมแปลภาษาตีความภาษาที่เขียนแปลงเป็นคำสั่งอินเตอร์มิเดียท



รูปที่ 3.7 การสร้างภาพแทนคำสั่งอินเตอร์มิเดียทผ่านทางเครื่องมือสร้างภาพมโนทัศน์

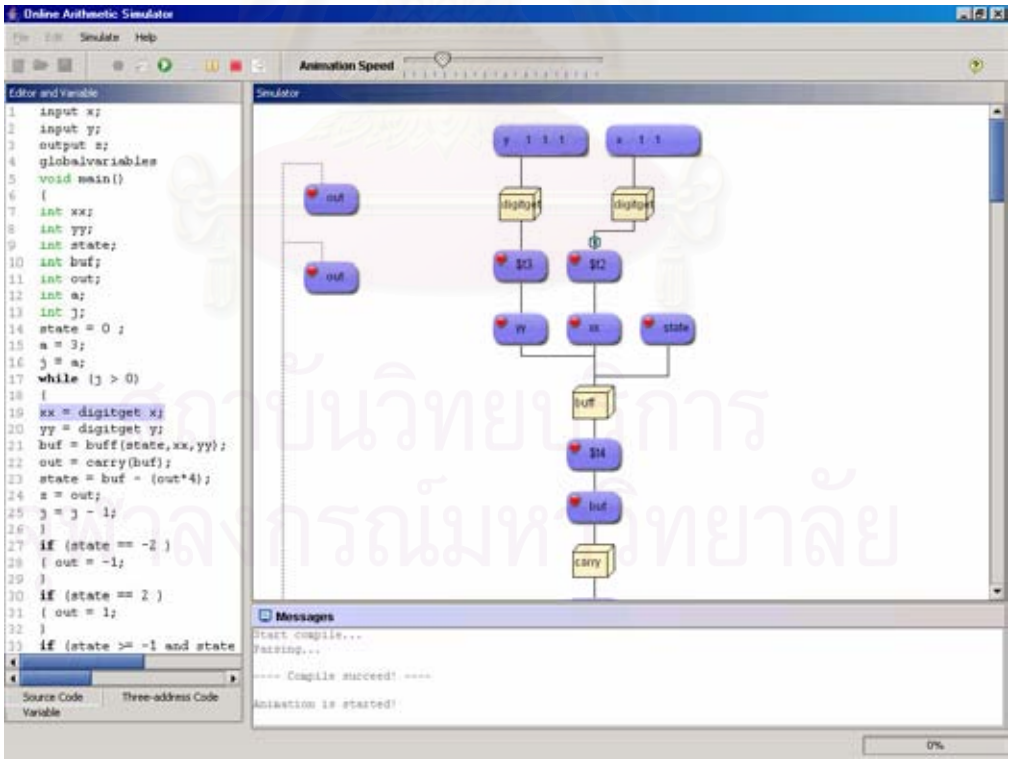
5. ขั้นตอนการแปลงคำสั่งให้เป็นกราฟิก

เป็นขั้นตอนหลังจากโปรแกรมแปลภาษาสร้างแฟ้มข้อมูลผลลัพธ์ในรูปแบบชุดคำสั่งอินเตอร์มิเดียท จากนั้นทำการสร้างภาพแทนคำสั่งอินเตอร์มิเดียทผ่านทางเครื่องมือสร้างภาพมโนทัศน์โดยใช้การแปลคำสั่งทีละคำสั่ง จากตัวอย่างอัลกอริทึมสามารถแปลงเป็นรูปภาพได้ดังรูปที่ 3.8

ตัวแปรในคำสั่งจะต้องระบุว่าเป็นตัวแปรประเภทใด เช่น ตัวแปรนำเข้า หรือตัวแปรผลลัพธ์ เป็นต้น จากนั้นสร้างเส้นความสัมพันธ์ระหว่างตัวแปรกับตัวดำเนินการ ตัวแปรนำเข้า และตัวแปรผลลัพธ์

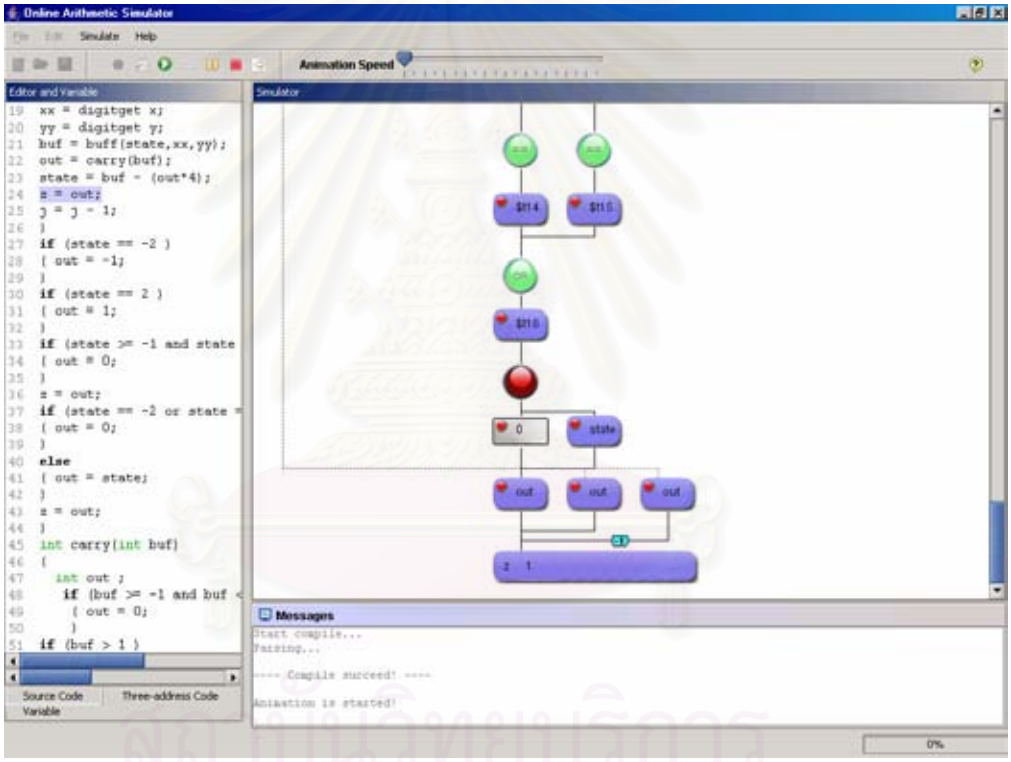
6. ขั้นตอนการแสดงภาพมโนทัศน์การทำงาน

การแสดงภาพมโนทัศน์การทำงานจะทำตามขั้นตอนการทำงานจากคำสั่งอินเตอร์มิเดียท เพื่อแสดงผลการเปลี่ยนแปลงของค่าตัวแปรที่ได้จากการคำนวณ จากตัวอย่างอัลกอริทึมการบวกแบบเชื่อมตรง มีตัวแปรนำเข้า คือ X และ Y โดยให้ค่านำเข้าคือ 110 และ 101 ตัวแปรผลลัพธ์คือ Z



รูปที่ 3.8 แสดงการนำค่าจากตัวแปรเข้ามาประมวลผล

การทำงานอัลกอริทึมจะรับข้อมูลครั้งละหนึ่งดิจิตทีละหลักจากตัวแปรนำเข้าทุกตัว ผ่านคำสั่งภายใน digitget โดยใช้ชื่อตัวแปรนำเข้าเป็นพารามิเตอร์ จากนั้นจะทำงานตามคำสั่งที่ระบุในอัลกอริทึม เช่น ตัวดำเนินการบวก และตรวจสอบเงื่อนไข ตามลำดับ จากนั้นนำผลลัพธ์ที่ได้เก็บไว้ที่ตัวแปร Z ตามลำดับของการคำนวณ โดยการทำงานจะรับค่าตัวแปรนำเข้านำมาประมวลผล และผลิตคำตอบอย่างต่อเนื่อง และทำการรับข้อมูลในดิจิตถัดไปนำเข้ามาทำงานจนกระทั่งหมดข้อมูลนำเข้า และผลิตคำตอบของผลลัพธ์ตัวแปร Z จะได้เท่ากับ 110-1 ในส่วนการควบคุมการทำงานสามารถทำงานเสร็จในขั้นตอนเดียว หรือทำงานทีละขั้นตอนได้ ลักษณะการแสดงภาพมโนทัศน์จะแสดงการเปลี่ยนแปลงของตัวแปรที่ใช้ในการทำงานตามลำดับการทำงานออกมาในรูปแบบการเคลื่อนไหวของตัวเลขก่อนและหลังจากการคำนวณ ดังรูปที่ 3.9

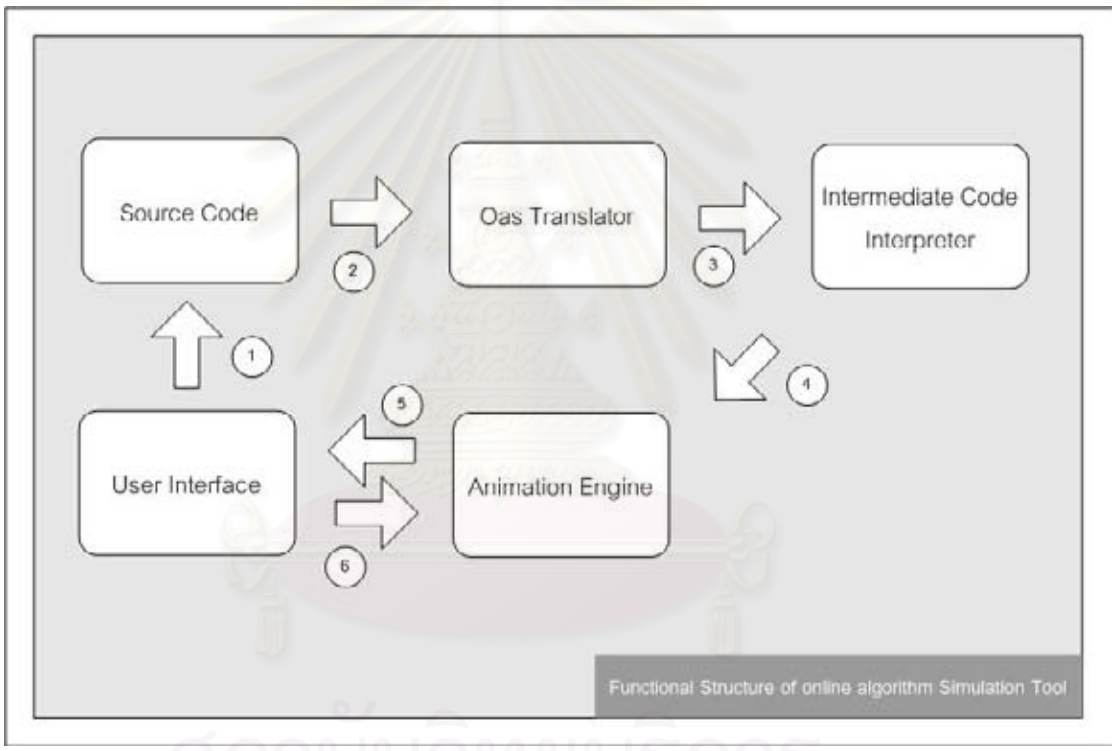


รูปที่ 3.9 แสดงการกำหนดค่าผลลัพธ์จากการคำนวน

บทที่ 4

การออกแบบเครื่องมือ

การออกแบบเครื่องมือสร้างภาพมโนทัศน์สำหรับผู้ออกแบบอัลกอริทึมการคำนวณแบบเชื่อมตรงระดับดิจิต เพื่อนำมาใช้เป็นเครื่องมือสำหรับช่วยสร้างภาพมโนทัศน์การทำงานของอัลกอริทึม โดยที่ผู้ออกแบบอัลกอริทึมสามารถตรวจสอบความถูกต้องของอัลกอริทึมและมองเห็นการเปลี่ยนแปลงค่าผลลัพธ์การทำงาน สามารถแสดงความสัมพันธ์ของตัวแปรที่เกี่ยวข้องและลำดับขั้นตอนการทำงานที่มีผลต่อตัวแปรนำเข้าและตัวแปรผลลัพธ์ ทางส่วนต่อประสานกับผู้ใช้

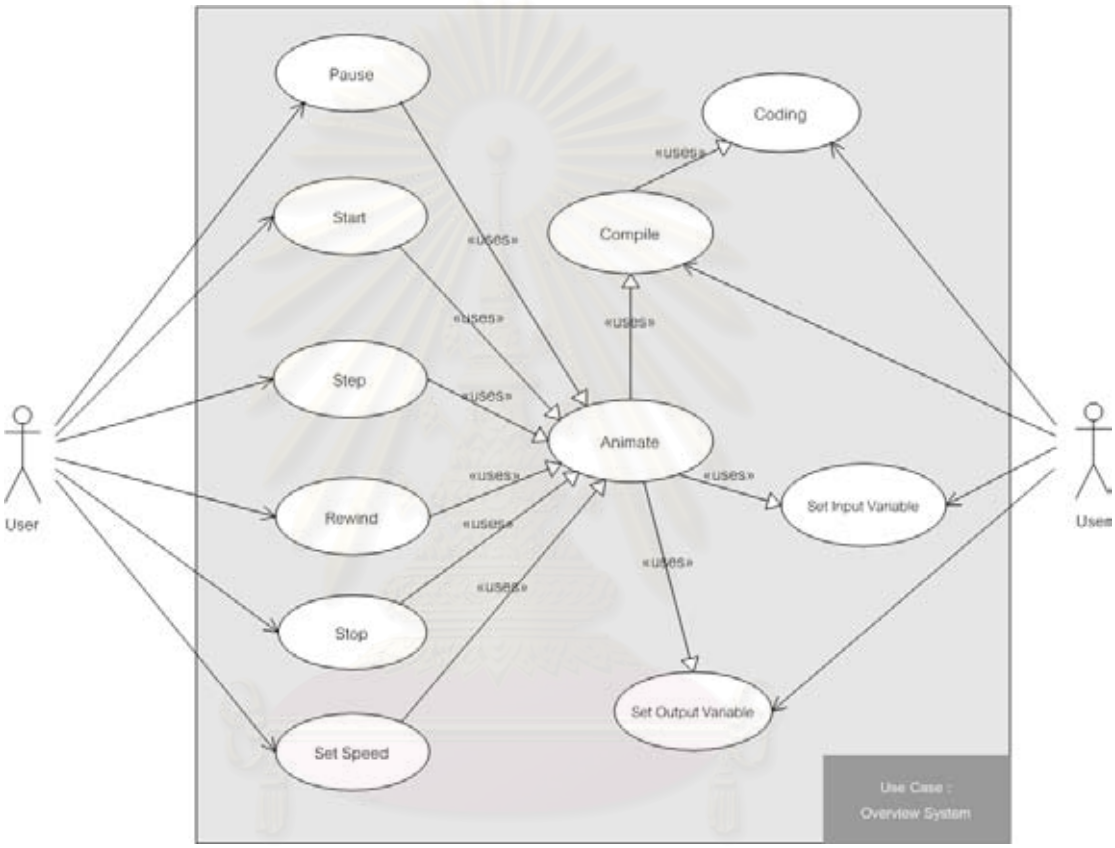


รูปที่ 4.1 โครงสร้างหน้าที่การทำงานของเครื่องมือสร้างภาพมโนทัศน์

โครงสร้างหน้าที่การทำงานของเครื่องมือสร้างภาพมโนทัศน์ ดังรูปที่ 4.1 ผู้ออกแบบอัลกอริทึมใช้ส่วนต่อประสานกับผู้ใช้ (User Interface) สำหรับใช้เขียนโปรแกรมต้นฉบับ หรือใช้แสดงและควบคุมการทำงานเครื่องมือสร้างภาพมโนทัศน์ โปรแกรมต้นฉบับถูกตรวจสอบความถูกต้อง และแปลงโปรแกรมต้นฉบับให้อยู่ในรูปแบบคำสั่งอินเตอร์มิเดียทผ่านตัวแปลภาษาส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ (Intermediate Code Interpreter) ทำหน้าที่ประมวลผลคำสั่งอินเตอร์มิเดียททีละบรรทัด เพื่อสร้างโครงสร้างข้อมูลและคำสั่งสำหรับสร้างภาพมโนทัศน์

เพื่อส่งให้กับส่วนประมวลผลการแสดงภาพมโนทัศน์ (Animation Engine) นำไปสร้างภาพโครงสร้างความสัมพันธ์ และแสดงภาพมโนทัศน์การทำงานของอัลกอริทึม

การออกแบบและพัฒนาเครื่องมือ จะอธิบายด้วยแผนภาพยูสเคส (Use Case Diagram) แผนภาพคลาส (Class Diagram) แผนภาพซีเควนซ์ (Sequence Diagram) โดยมีรายละเอียดดังนี้



รูปที่ 4.2 ยูสเคสภาพรวมเครื่องมือสร้างภาพมโนทัศน์

การใช้งานเครื่องมือจะต้องใช้ (Use) กิจกรรมการทำงานของเครื่องมือสร้างภาพมโนทัศน์โดยมีส่วนประกอบดังนี้

1) ผู้ใช้หรือแอคเตอร์ (Actors) หมายถึง ผู้ออกแบบอัลกอริทึม

2) ยูสเคส Coding หมายถึง ผู้ออกแบบอัลกอริทึมเขียนโปรแกรมต้นฉบับด้วยอัลกอริทึมการคำนวณแบบเชื่อมตรงระดับดิจิตสำหรับแสดงภาพมโนทัศน์

3) ยูสเคส set input variable หมายถึง ผู้ออกแบบอัลกอริทึมป้อนตัวแปรนำเข้าเพื่อใช้ในการคำนวณค่า

4) ยูสเคส set output variable หมายถึง ผู้ออกแบบอัลกอริทึมป้อนตัวแปรผลลัพธ์เพื่อใช้แสดงผลลัพธ์จากการคำนวณ

5) ยูสเคส Compile หมายถึง ผู้ออกแบบอัลกอริทึมตรวจสอบความถูกต้องของโปรแกรมต้นฉบับ โดยใช้ยูสเคส Coding

6) ยูสเคส Animate หมายถึง การแสดงภาพมโนทัศน์

7) ยูสเคส Start หมายถึง สั่งแสดงภาพมโนทัศน์

8) ยูสเคส Pause หมายถึง หยุดแสดงภาพมโนทัศน์ชั่วคราว

9) ยูสเคส Step หมายถึง สั่งแสดงภาพมโนทัศน์การทำงานครั้งละ 1 คำสั่ง

10) ยูสเคส Stop หมายถึง หยุดแสดงภาพมโนทัศน์

11) ยูสเคส Rewind หมายถึง เริ่มแสดงภาพมโนทัศน์ใหม่

12) ยูสเคส Change speed หมายถึง การปรับเปลี่ยนความเร็วการแสดงภาพมโนทัศน์

ขั้นตอนการทำงานโดยรวมของเครื่องมือสร้างภาพมโนทัศน์ อธิบายด้วยแผนภาพซีเควนซ์ ดังรูปที่ 4.3 – 4.6 เริ่มจากผู้ออกแบบอัลกอริทึม กำหนดค่าเริ่มต้นการทำงานคือ ป้อนโปรแกรมต้นฉบับเขียนด้วยอัลกอริทึมการทำงานแบบเชื่อมตรงระดับดิจิต, กำหนดตัวแปรนำเข้าและตัวแปรผลลัพธ์ ผ่านทางส่วนประสานงานผู้ใช้ ชนิดตัวแปรนำเข้าสามารถเลือกได้ 2 รูปแบบ คือจำนวนซ้ำซ้อน (Redundant Nymber) และจำนวนไม่ซ้ำซ้อน (Non-Redundant Number)

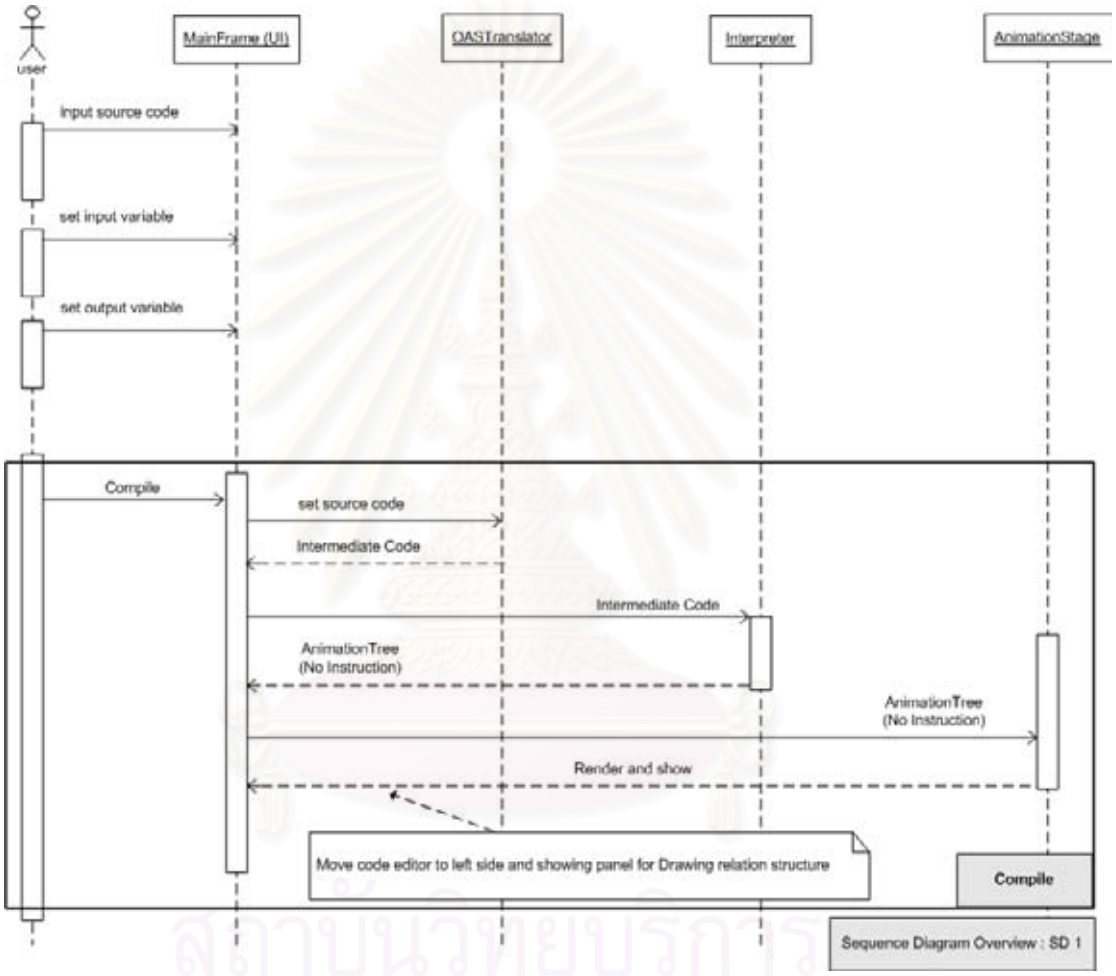
1) จำนวนซ้ำซ้อน คือ ตัวแปรนำเข้าที่สามารถแสดงค่าได้มากกว่าหนึ่งรูปแบบในงานวิจัยนี้ใช้ตัวแปรนำเข้าในรูปแบบดิจิตของเลขฐานสองแบบมีเครื่องหมาย เช่น ผู้ออกแบบอัลกอริทึมป้อนค่า 5 ฐานสิบ โปรแกรมจะทำการแปลงให้อยู่ในรูปแบบเลขฐานสองคือ 101 ผู้ออกแบบอัลกอริทึมสามารถแก้ไขรูปแบบการแสดงค่าของตัวแปรนำเข้าโดยใช้ $(101)_2$ หรือ $(10\text{-}1\text{-}1)_2$ ซึ่งจะได้ค่าเท่ากับ 5 ฐานสิบเช่นเดียวกัน

2) จำนวนไม่ซ้ำซ้อน คือ ตัวแปรนำเข้าที่สามารถแสดงค่าได้เพียงรูปแบบเดียวโดยค่าที่สามารถป้อนได้คือเลข 0 หรือ 1 เท่านั้น

การสร้างภาพมโนทัศน์หลักการทำงานคือ การพิจาราณาค่าของตัวแปรนำในรูปแบบเลขดิจิตฐานสอง โดยเริ่มจากหลักด้านซ้ายซึ่งมีเลขนัยสำคัญสูงสุด จะถูกดึงค่าครั้งละหนึ่งดิจิตหรือทีละหลักของทุกตัวแปรนำเข้า โดยจะนำมาใช้คำนวณพร้อมกันจนถึงลำดับที่มีเลขนัยสำคัญต่ำสุด ค่าที่เหลือจากการคำนวณในแต่ละหลักจะต้องรอตัวทดจากหลักถัดไปเพื่อทยอยผลิต

ค่าคำตอบของตัวแปรผลลัพธ์ออกมาทีละดิจิตตามลำดับ ซึ่งรูปแบบของรูปภาพที่แสดง ขึ้นอยู่กับโปรแกรมต้นฉบับตามที่อัลกอริทึมได้กำหนดไว้

การทำงานเริ่มจากผู้ออกแบบอัลกอริทึมตรวจสอบความถูกต้องของโปรแกรมต้นฉบับด้วย OASTranslator ผ่านการสั่ง “Compile” โดยใช้โปรแกรมต้นฉบับเป็นข้อมูลนำเข้า เพื่อแปลงโปรแกรมต้นฉบับเป็นคำสั่งอินเตอร์มิเดียท ดังรูปที่ 4.3

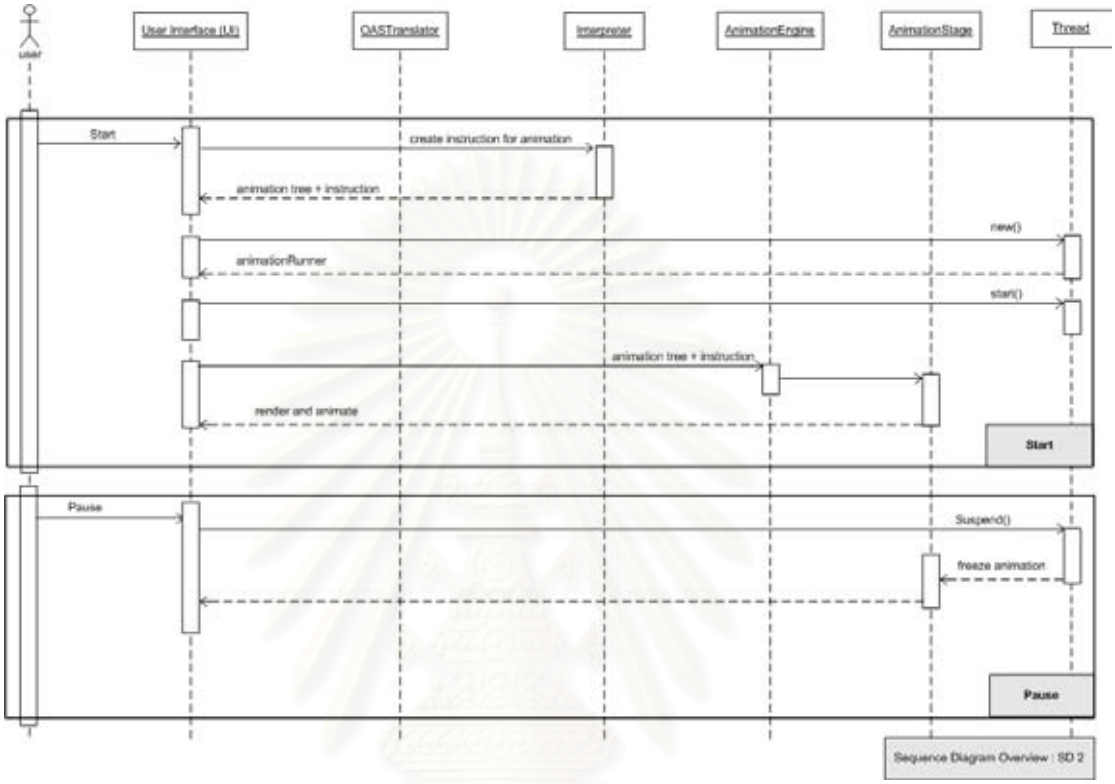


รูปที่ 4.3 แผนภาพซีเควนซ์แสดงการทำงานโดยรวมของเครื่องมือสร้างภาพมโนทัศน์

ส่วนประสานงานกับผู้ใช้จะส่งคำสั่งอินเตอร์มิเดียทโค้ดให้ส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ซึ่งเรียกใช้ผ่านทาง Interpreter โดยขั้นตอนนี้จะสร้างโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์นำไปแสดงที่ AnimationStage หรือส่วนแสดงภาพมโนทัศน์

ผู้ออกแบบอัลกอริทึมเลือกแสดงภาพมโนทัศน์ผ่านคำสั่ง “Start” ขั้นตอนนี้จะเรียกใช้โครงสร้างข้อมูลร่วมกับคำสั่งสำหรับแสดงภาพมโนทัศน์ทั้งหมดส่งให้ AnimationEngine เพื่อ

นำไปแสดงการทำงานของอัลกอริทึมโดยใช้ภาพเคลื่อนไหวและการเปลี่ยนแปลงค่าของข้อมูลผ่านทาง AnimationStage ซึ่งติดต่อกับส่วนประสานงานกับผู้ใช้ภายใต้การทำงานของเทรด (Thread) ช่วยในการแบ่งแยกการทำงานของโปรเซสในระหว่างการแสดงภาพมโนทัศน์ ดังรูปที่ 4.3 (ก)

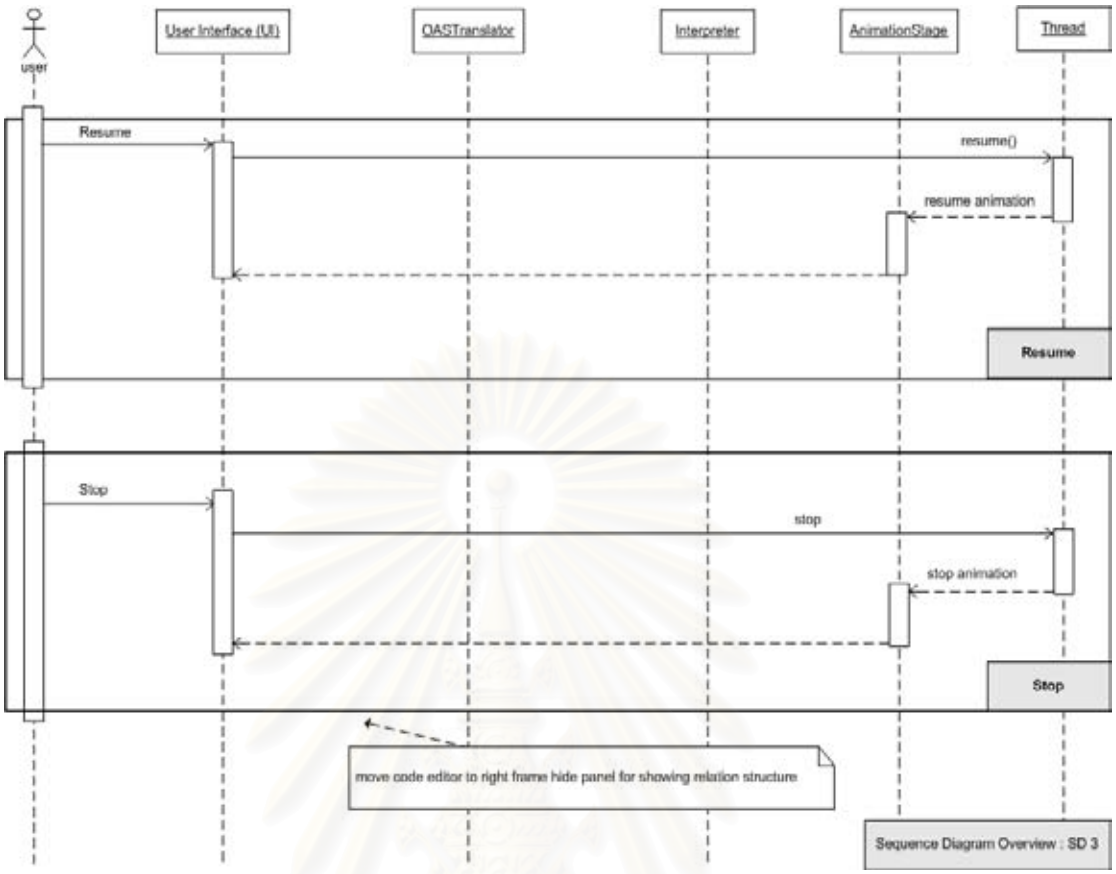


รูปที่ 4.3 (ก) แผนภาพซีเควนซ์แสดงการทำงานโดยรวมของเครื่องมือสร้างภาพมโนทัศน์

การสั่งหยุดการทำงาน (Pause) ผู้ใช้งานสามารถหยุดการทำงานในระหว่างแสดงภาพมโนทัศน์ผ่านเมทธอด suspend () ไปยังเทรดเพื่อหยุดการแสดงภาพมโนทัศน์ ดังรูปที่ 4.3 (ก)

การสั่งการทำงานต่อเนื่อง (Resume) ผู้ใช้งานสามารถสั่งให้แสดงภาพมโนทัศน์ต่อเนื่องภายหลังจากสั่งคำสั่งหยุดการแสดงภาพมโนทัศน์ชั่วคราวผ่านเมทธอด resume () ดังรูปที่ 4.3 (ข)

การสั่งหยุดการทำงาน (Stop) ผู้ใช้งานสามารถสั่งให้หยุดแสดงการทำงานไปที่ AnimationEngine โดยที่ AnimationEngine จะทำหน้าที่สั่งหยุดการทำงานกลับไปที่ส่วนประสานงานผู้ใช้ และย้ายส่วนการเขียนโปรแกรมต้นฉบับกลับสู่สถานะรอรับการแก้ไขโปรแกรมต้นฉบับทางหน้าต่างทางขวามือของผู้ใช้งาน ดังรูปที่ 4.3 (ข)



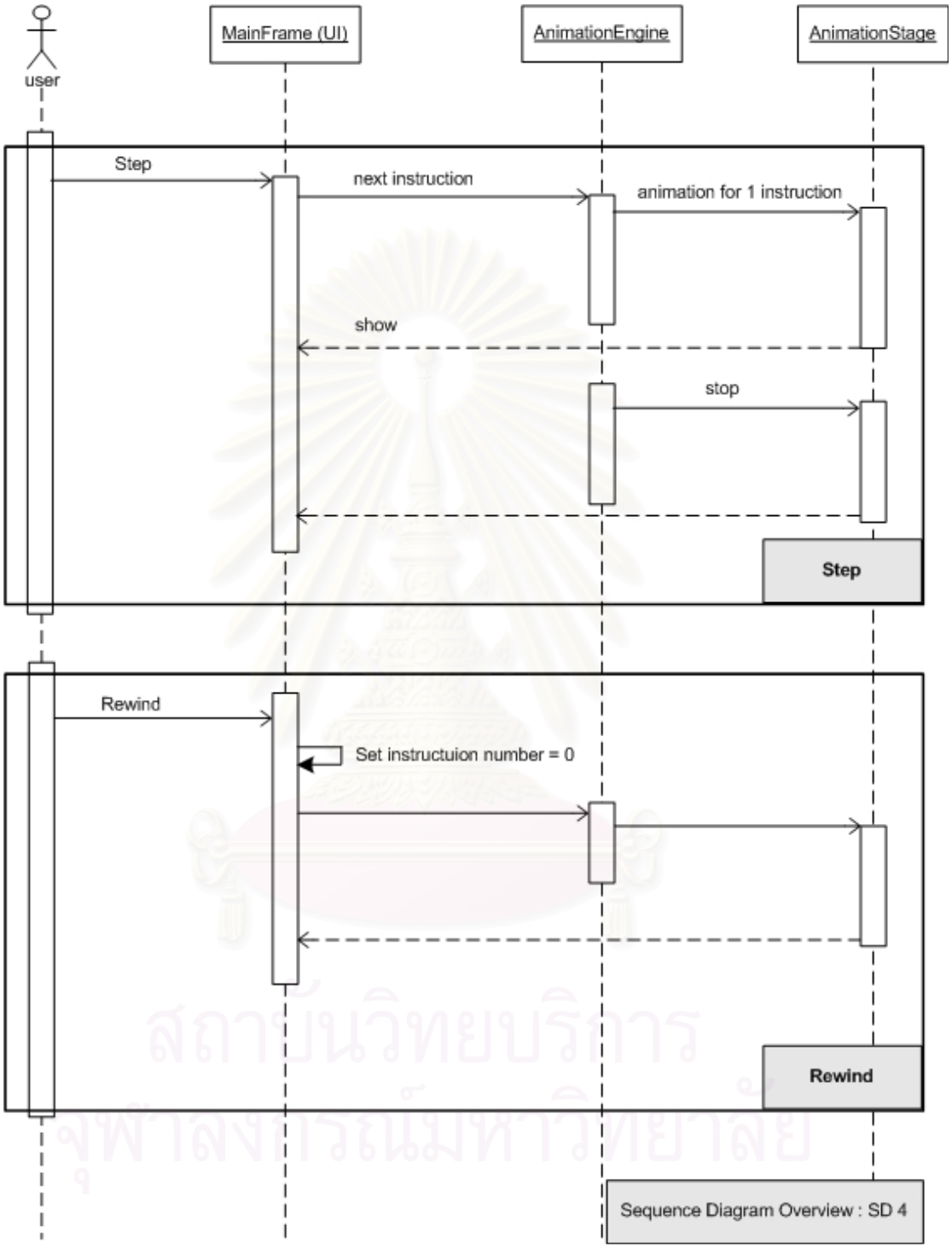
รูปที่ 4.3 (ข) แผนภาพซีเควนซ์แสดงการทำงานโดยรวมของเครื่องมือสร้างภาพมโนทัศน์

การสั่งทำงานครั้งละคำสั่ง (Step) ผู้ใช้งานสามารถสั่งให้แสดงขั้นตอนการทำงาน ครั้งละหนึ่งคำสั่งไปที่ AnimationEngine โดยที่ AnimationEngine จะทำหน้าสั่งให้มีการแสดงทำงาน ซึ่งจะแสดงในส่วน AnimationStage แสดงกลับไปยังส่วนประสานงานผู้ใช้จนครบคำสั่งก็จะสั่งหยุดการทำงานเพื่อรอคำสั่งจากผู้ใช้งานต่อไป ดังรูปที่ 4.3 (ค)

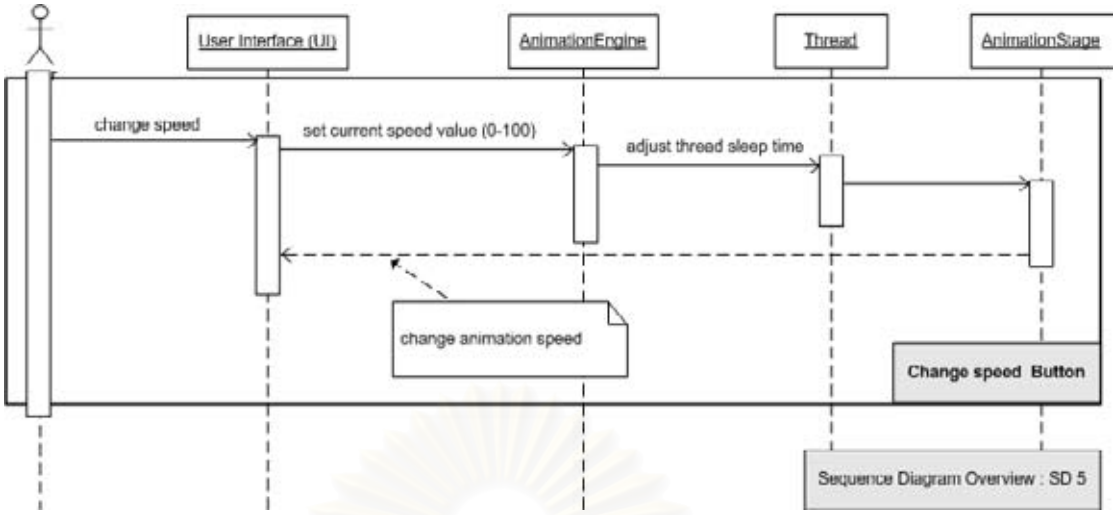
การสั่งเริ่มแสดงภาพมโนทัศน์ใหม่ (Rewind) ผู้ใช้งานสามารถสั่งให้เริ่มแสดงภาพมโนทัศน์การทำงานใหม่โดยจะกำหนดลำดับไปที่จุดเริ่มต้นของคำสั่งภายในเทรดที่กำลังทำงานผ่านทาง AnimationEngine ให้เริ่มการแสดงภาพมโนทัศน์ใหม่ ไปที่ ดังรูปที่ 4.3 (ค)

การสั่งปรับเปลี่ยนความเร็วการแสดงภาพมโนทัศน์ (Change speed) ผู้ใช้งานสามารถสั่งปรับเปลี่ยนความเร็วในการแสดงภาพมโนทัศน์ เพื่อติดตามการทำงานให้ส่วนที่ต้องการได้ชัดเจนยิ่งขึ้นโดยการส่งค่าช่วงของความเร็วไปที่ AnimationEngine จากนั้น AnimationEngine จะ

ปรับปรุงความเร็วไปที่ AnimationStage และแสดงผลกลับไปยังผู้ใช้ทางส่วนประสานงานผู้ใช้ ดังรูปที่ 4.3 (ง)



รูปที่ 4.3 (ค) แผนภาพซีเควนซ์แสดงการทำงานโดยรวมของเครื่องมือสร้างภาพมโนทัศน์



รูปที่ 4.3 (ง) แผนภาพซีเควนซ์แสดงการทำงานโดยรวมของเครื่องมือสร้างภาพมโนทัศน์

การออกแบบเครื่องมือสร้างภาพมโนทัศน์แบ่งขั้นตอนการทำงานออกเป็น 2 ส่วน คือ

4.1 ส่วนการแปลภาษา

4.2 ส่วนการสร้างภาพมโนทัศน์

4.1 ส่วนการแปลภาษา (Language Translation)

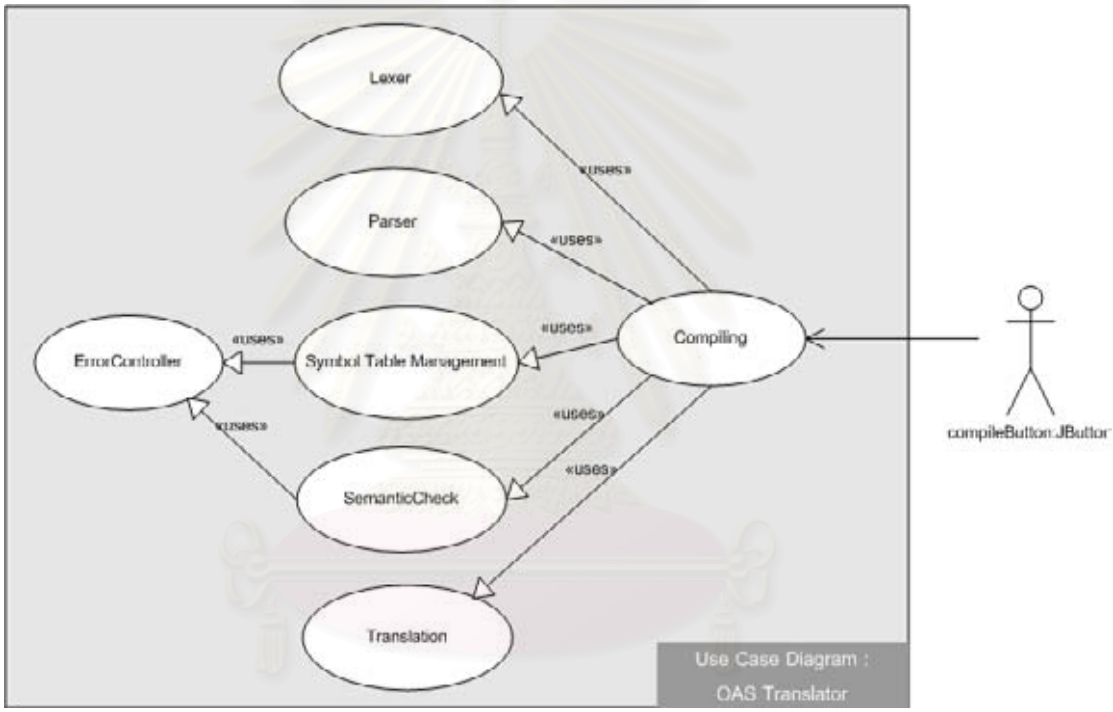
เป็นส่วนสำหรับตรวจสอบความถูกต้องของโปรแกรมต้นฉบับ และแปลงโปรแกรมต้นฉบับเป็นคำสั่งอินเตอร์มิเดียทโดยใช้รูปแบบทรีแอดเดรสโค้ด (Three-address code) จากรูปภาพที่ 4.4 แผนภาพยูสเคสตัวแปลภาษาสำหรับใช้แสดงภาพมโนทัศน์ แสดงกิจกรรมต่างๆ ที่เกิดขึ้นในขั้นตอนการแปลภาษา โดยจะถูกเรียกใช้จากส่วนต่อประสานงานกับผู้ใช้ เมื่อ ActionListener ตรวจพบการสั่งคอมไพล์โปรแกรมต้นฉบับเพื่อตรวจสอบความถูกต้อง แอคเตอร์ใช้ยูสเคส OASTranslator ซึ่งจะต้องใช้ (Use) กิจกรรมการทำงานของตัวแปลภาษาโอเอเอส โดยมีส่วนประกอบดังนี้

- แอคเตอร์ หมายถึง ผู้ใช้งานสั่งคอมไพล์โปรแกรมต้นฉบับผ่านทางส่วนประสานงานผู้ใช้
- ยูสเคส Lexer หมายถึง ส่วนการวิเคราะห์คำ (Lexical Analyzer) ซึ่งอ่านโปรแกรมต้นฉบับและสร้างชุดของกระแสโทเค็นและตรวจสอบความถูกต้องของคำที่นิยามในภาษา

- ยูสเคส Parser หมายถึง ส่วนการวิเคราะรูปประโยค (Syntax Analyzer) สำหรับใช้ตรวจสอบว่าโทเค็นต่างๆ ที่ได้จากยูสเคส Lexer เรียงตัวถูกต้องตามหลักไวยากรณ์ของภาษาหรือไม่เพื่อนำไปสร้างไวยากรณ์ต้นไม้นามธรรม (Abstract Syntax Tree)

- ยูสเคส Symbol Table Management หมายถึง ตารางสัญลักษณ์ ใช้จัดเก็บข้อมูลของตัวแปรและขอบเขตการทำงานของตัวแปร

- ยูสเคส SemanticCheck คือ ส่วนการวิเคราะห์ความหมาย (Semantic Analyzer) ใช้ตรวจสอบความถูกต้องของชนิดข้อมูล และตรวจจับข้อผิดพลาดของโปรดักชันในขั้นตอนการออกแบบไวยากรณ์ (Error Production) ซึ่งจะอธิบายรายละเอียดในขั้นตอนถัดไป



รูปที่ 4.4 แผนภาพยูสเคสตัวแปลภาษา

- ยูสเคส Translation คือ ส่วนการแปลงโปรแกรมต้นฉบับเป็นคำสั่งอินเตอร์มิเดียทและจัดเก็บไว้ในแฟ้มข้อมูลคำสั่งอินเตอร์มิเดียท

- ยูสเคส ErrorController คือ การตรวจจับข้อผิดพลาด (Error Handing) ในขั้นตอนการแปลโปรแกรมต้นฉบับ

4.1.1 การออกแบบภาษา (Language Design)

จุดมุ่งหมายหลักของการออกแบบภาษา เพื่อกำหนดรูปแบบและข้อจำกัดของภาษาสำหรับผู้ออกแบบอัลกอริทึม เพื่อใช้ทดสอบอัลกอริทึม โดยที่โครงสร้างของภาษามีลักษณะไม่ซับซ้อนสามารถทำความเข้าใจและศึกษาได้ง่าย โดยใช้ชื่อเรียกภาษาที่พัฒนานี้ว่า ภาษาโอเอเอส (OAS Language) มีรูปแบบ ดังนี้

1) โครงสร้างทั่วไปของภาษาโอเอเอสประกอบด้วยส่วนต่างๆ ดังนี้

- ส่วนการประกาศตัวแปรนำเข้า
- ส่วนการประกาศตัวแปรผลลัพธ์
- ส่วนการประกาศตัวแปรส่วนกลาง
- ส่วนการประกาศฟังก์ชันหลักและยูสเซอร์ฟังก์ชัน

ตารางที่ 4.1 โครงสร้างภาษา

โครงสร้างทางภาษา	โปรแกรมต้นฉบับ

Input Declaration
(ส่วนการประกาศตัวแปรนำเข้า)
▼
Output Declaration
(ส่วนการประกาศตัวแปรผลลัพธ์)
▼
Global Variable Declaration
(ส่วนการประกาศตัวแปรส่วนกลาง)
▲
globalvariables
Main Function and User Functions
(ส่วนการประกาศฟังก์ชันหลักหรือยูสเซอร์ฟังก์ชัน)
{
▼
}
	input redundant x;
	input redundant y;
	output z;
	int a;
	Globalvariables
	void main()
	{
	int k;
	k = 1 + 1 ;
	k = k + a ;
	if (k == 1)
	{
	z = k ;
	}
	}

ข้อกำหนดของภาษาสำหรับให้ผู้ออกแบบอัลกอริทึมใช้เขียนโปรแกรมแสดงการทำงานของอัลกอริทึมการคำนวณแบบเชื่อมตรง มีดังต่อไปนี้

1. ชนิดข้อมูลที่ใช้ในภาษา ประกอบด้วย ชนิดจำนวนเต็ม, ชนิดบลูลีน, ชนิดสายอักขระบิต ชนิดจำนวนเต็มค่าที่เป็นไปได้อยู่ในช่วง -2,147,483,648 ถึง -2,147,483,647 ข้อมูลชนิดบลูลีนจะมีค่าที่เป็นไปได้คือ “True” หรือ “False” ข้อมูลชนิดสายอักขระบิตประกอบด้วยค่า 1 หรือ 0 เท่านั้น

2. ฟังก์ชันที่ใช้มีทั้งแบบส่งค่ากลับและแบบไม่ส่งค่ากับ

3. พารามิเตอร์ที่ใช้ในฟังก์ชันใช้รูปแบบการส่งผ่านโดยค่า (Pass by value)

4. ตัวแปรแถวลำดับใช้เพียง 1 มิติ

5. ไม่สนับสนุนการเรียกใช้งานแบบฟังก์ชันเรียกซ้ำ (Recursive Function) และการประกาศฟังก์ชันแบบซ้อนทับ (Nested Function Declaration)

6. ภายในโปรแกรมต้นฉบับต้องมีฟังก์ชันอย่างน้อยชื่อ “Main“ เป็นฟังก์ชันหลัก (Main function) สำหรับใช้เป็นจุดเริ่มต้นการทำงาน

7. คำสั่งทางเลือกคำหลัก “if” สามารถมี else บล็อกได้เพียง 1 บล็อกไม่สนับสนุนคำสั่งการเรียกใช้แบบ “else if”

8. การประกาศตัวแปรส่วนกลางประกาศอยู่เหนือคำหลัก globalvariable

9. การประกาศตัวแปรเฉพาะที่ประกาศภายใต้ยูสเซอร์ฟังก์ชัน

10. คำสั่งวนซ้ำใช้คำหลัก While จะวนซ้ำจนกระทั่งนิพจน์เงื่อนไขเป็นจริง

11. เครื่องหมาย “;” ใช้เป็นตัวจบประโยคคำสั่ง

12. คำอธิบาย (Comments) ใช้เครื่องหมาย /* …. ข้อความ .... * / เป็นวิธีการละข้อความสำหรับข้อความขนาดยาว และเครื่องหมาย “//” เพื่อละข้อความที่มีความยาวไม่เกิน 1 บรรทัด

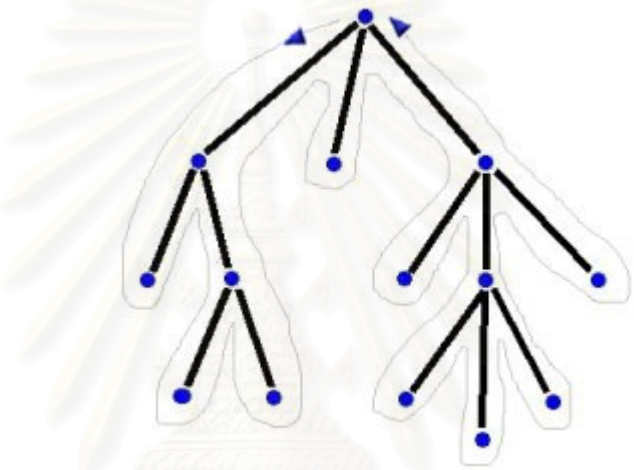
13. คำสั่งภายในจัดเป็นคำสงวน

14. ชื่อตัวแปรที่ใช้งานเป็นรูปแบบไวต่ออักษรเล็กใหญ่

การอธิบายรูปแบบกฎเกณฑ์ของภาษาโปรแกรมใช้ส่วนขยายของอีเอ็นเอฟ (EBNF: Extend Backus Norm Form) รายละเอียดแสดงในภาคผนวก ก. รูปแบบการเขียนโปรแกรมรายละเอียดแสดงในภาคผนวก ค.

ในงานวิจัยนี้ได้ใช้ขั้นตอนการทำงานส่วนหน้า (Front End) เพื่อสร้างตัวแปลภาษา โดยแบ่งขั้นตอนการทำงานออกเป็นสี่ขั้นตอนดังนี้ การวิเคราะห์คำ, การวิเคราะห์รูปประโยค, การวิเคราะห์ความหมาย, การสร้างคำสั่งอินเตอร์มิเดียท เพื่อให้ได้ผลลัพธ์คือ คำสั่งอินเตอร์มิเดียท นำไปแสดงภาพมโนทัศน์การทำงาน

ผู้วิจัยเลือกใช้ SableCC [15] เป็นเครื่องมือพัฒนาคอมไพเลอร์แบบโครงสร้างเชิงวัตถุ เพื่อใช้สร้างตัวแปลภาษา โดยการนิยามคำและไวยากรณ์ (รายละเอียดแสดงในภาคผนวก ก.) เพื่อสร้างส่วนการวิเคราะห์คำ และส่วนการวิเคราะห์รูปประโยค



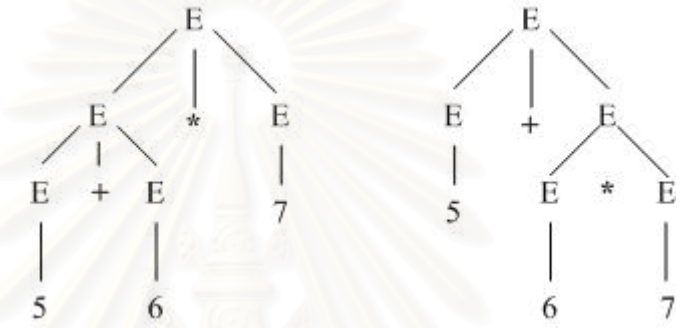
รูปที่ 4.5 การเข้าถึงไวยากรณ์ต้นไม้ตามแนวลึก

นอกจากส่วนการวิเคราะห์คำ และส่วนการวิเคราะห์รูปประโยค โครงสร้างเชิงวัตถุจะถูกนำมาสร้างส่วนการจัดการกับตารางสัญลักษณ์ สำหรับใช้ในการจัดเก็บและอ้างอิงขอบเขตของตัวแปร การสร้างส่วนวิเคราะห์ความหมาย และการสร้างชุดคำสั่งอินเตอร์มิเดียท โดยใช้การสืบทอดคุณสมบัติจากคลาส Tree-walker ซึ่งถูกสร้างมาพร้อมกับโครงสร้างเชิงวัตถุพัฒนาด้วย SableCC ทำหน้าที่ ท่องโหนดไวยากรณ์ต้นไม้โดยใช้ extended version of the visitor design pattern ทำให้สามารถเพิ่มแอคชันโค้ด (Action Code) หรือโค้ดเฉพาะที่ผู้พัฒนาตัวแปลภาษาต้องการให้โปรแกรมแปลภาษาทำงานอย่างไร เมื่อเข้าหรือออกจากโหนดในไวยากรณ์ต้นไม้นามธรรม โดยวิธีการเข้าถึงแบบกำหนดลำดับทำให้สามารถเข้าถึงโหนดทุกโหนดในไวยากรณ์ต้นไม้นามธรรมในลักษณะการเข้าถึงตามแนวลึก (Depth-First Traversal) หรือการเข้าถึงย้อนกลับตามแนวลึก (Reverse Depth-fiest) สำหรับงานวิจยนี้เลือกการเข้าถึงตามแนวลึก โดยการท่องโหนดจะเริ่มจากโหนดรากผ่านโหนดทุกโหนดในไวยากรณ์ต้นไม้นามธรรมตามแนวลูกศร

ดังรูปที่ 4.5 โดยสืบทอดคุณสมบัติจากคลาส DepthFirstAdapter ด้วยการเพิ่มแอคชันโค้ด (Action Code) ลงในโอเวอร์ไรด์เมทธอด (Override Method)

ความคลุมเครือ (Ambiguity)

การบรรยายไวยากรณ์การคำนวณทางคณิตศาสตร์บางครั้งมีความคลุมเครือ เนื่องมาจากสามารถสร้างไวยากรณ์ต้นไม้ได้มากกว่า 1 รูปแบบ ลักษณะเช่นนี้เป็นสิ่งที่ต้องหลีกเลี่ยงไม่ให้เกิดขึ้นในขั้นตอนการบรรยายไวยากรณ์ ดังรูปที่ 4.6



รูปที่ 4.6 ลักษณะไวยากรณ์ต้นไม้ที่มีความคลุมเครือ

ความแตกต่างสำหรับนิพจน์ “5+6*7” คือ “(5+6) * 7” หรือ “5 + (6*7)” วิธีแก้ไขปัญหาโดยใช้กฎการกำหนดลำดับการทำงานของตัวดำเนินการ (Precedence of Operators) [8] [16] โดยการเพิ่มไวยากรณ์ในส่วนของโปรดักชัน Factor และเปลี่ยนตัวดำเนินการที่มีลำดับสูงสุดนำไปไว้ในส่วนท้ายสุดของไวยากรณ์เรียงตามลำดับ ดังนี้

expression ⟶ expression **plus** term |
expression **minus** term |
term;

term ⟶ term **mult** not_operator |
term **div** not_operator |
not_operator;

not_operator ⟶ **not** factor |
factor;

factor ⟶ **l_parenthese** condition **r_parenthese** |
value;

ขอบเขต (Scope)

การประกาศใช้ตัวแปรชื่อเดียวกันใช้ส่วนต่างๆของโปรแกรมต้นฉบับ ชนิดขอบเขต (Scope Type) เป็นตัวกำหนดการประกาศชื่อตัวแปรจะถูกนำมาใช้ในส่วนใดของโปรแกรมต้นฉบับ ชื่อตัวแปรที่ประกาศภายในฟังก์ชันเรียกว่าเป็นตัวแปรแบบเฉพาะที่ อยู่ภายใต้เครื่องหมาย { } นอกจากนั้นแล้วจะเป็นตัวแปรส่วนกลางประกาศในส่วนเหนือคำหลัก globalvariables ซึ่งสามารถอ้างอิงได้จากทุกขอบเขตภายในโปรแกรมต้นฉบับ ในขณะแปลโปรแกรมต้นฉบับ ตารางสัญลักษณ์จะถูกนำมาใช้ในตรวจสอบขอบเขตของตัวแปร การเลือกขอบเขตในงานวิจัยนี้เลือกขอบเขตสถิตแบบไม่ซ้อนกัน (Static Scope without Nested Procedures)

การทำงานในตัวแปลภาษาอาจเกิดข้อผิดพลาดขึ้นในแต่ละขั้นตอน เช่น ส่วนการวิเคราะห์คำ, ส่วนการวิเคราะห์รูปประโยค, ส่วนการวิเคราะห์ความหมาย ภายหลังจากที่ได้ตรวจพบข้อผิดพลาดขึ้นแล้วต้องมีวิธีแสดงข้อผิดพลาดให้กับผู้ใช้ได้รับทราบและจัดการกับข้อผิดพลาด ข้อผิดพลาดที่สามารถเกิดขึ้นได้เช่น ส่วนการวิเคราะห์คำ ได้แก่ ตัวดำเนินการ, คำสงวน, โทเค็นไม่ถูกต้อง ส่วนการวิเคราะห์รูปประโยค ได้แก่ โทเค็นในประโยคคำสั่งเรียงตัวไม่ถูกต้องตามหลักไวยากรณ์ ส่วนการวิเคราะห์ความหมาย ได้แก่ กำหนดค่าให้ตัวแปรต่างชนิดกัน เป็นต้น

ในงานวิจัยนี้ได้ใช้การตรวจสอบข้อผิดพลาดจากคลาส LexerException ตรวจจับข้อผิดพลาดของตัววิเคราะห์คำ และคลาส ParserException ตรวจจับข้อผิดพลาดของการตรวจสอบรูปประโยคจาก SableCC และสร้างการตรวจจับข้อผิดพลาดของโปรดักชัน (Error productions) [16] คือ การปรับปรุงไวยากรณ์ที่พบว่าอาจจะเกิดข้อผิดพลาดขึ้นได้ง่าย โดยใช้วิธีเลือกโปรดักชันที่ต้องการและทำการแบ่งออกเป็นโปรดักชันทางเลือกหลายๆ โปรดักชัน เพื่อทำหน้าที่ในการแก้ไขข้อผิดพลาดเป็นการเฉพาะ จากนั้นนำไวยากรณ์ที่ปรับปรุงแล้วไปสร้างตัวแปลภาษา ก็จะทำให้ตัวแปลภาษาสามารถจัดการกับข้อผิดพลาดได้ดียิ่งขึ้น โดยการปรับปรุงไวยากรณ์ในส่วนต่างๆ ในไวยากรณ์ของ SableCC ดังนี้

- การตรวจสอบตัวดำเนินการการกำหนดค่า

การกำหนดค่าอาจจะพบข้อผิดพลาดในการใช้เครื่องหมายกำหนดค่าที่ผิดไปจากไวยากรณ์ที่กำหนดไว้ เช่นเครื่องหมายมีลักษณะเหมือนกับการกำหนดค่าในภาษาอื่น ดังนั้นจึงเลือกเครื่องหมายที่อาจจะเกิดข้อผิดพลาดในการกำหนดค่ามาปรับปรุงไวยากรณ์การตรวจสอบข้อผิดพลาด

โดยการเพิ่มโปรดักชันทางเลือก {incorrect1} สำหรับตรวจสอบเครื่องหมาย '==' และ โปรดักชันทางเลือก {incorrect2} สำหรับเครื่องหมาย ':=' ถ้าหากพบขณะท่องโหนดไวยากรณ์ต้นไม้แสดงว่าพบข้อผิดพลาดในส่วนของตัวดำเนินการการกำหนดค่า

assignment = {correct} variable assign type_value|
{incorrect1} variable equal type_value|
{incorrect2} variable colonequal type_value;

- เครื่องหมายแบ่งพารามิเตอร์สำหรับฟังก์ชัน

การประกาศฟังก์ชันและการเรียกใช้ฟังก์ชัน โดยปกติใช้เครื่องหมายแบ่งแยกคือเครื่องหมาย "," หรือ เครื่องหมาย ";" ในไวยากรณ์ภาษาที่นิยามขึ้นใช้เครื่องหมาย "," สำหรับแบ่งแยกพารามิเตอร์ ส่วนนี้จะปรับปรุงโดยการเพิ่มโปรดักชันทางเลือก {incorrect} สำหรับตรวจสอบเครื่องหมาย ";"

function_return_value identifier l_parenthese declarator_parameters? r_parenthese function_block;
declarator_parameters = data_type identifier more_declarator_parameters? ;
more_declarator_parameters =
{correct} comma declarator_parameters |
{ incorrect} semicolon declarator_parameters ;

function_call = identifier l_parenthese function_call_parameters? r_parenthese;
function_call_parameters = function_call_value more_function_call_parameters?;
more_function_call_parameters =
{correct} comma function_call_parameters |
{ incorrect} semicolon function_call_parameters ;

builtin_parameters = builtin_call_value more_builtin_call_parameters?;
more_builtin_call_parameters =
{correct} comma builtin_parameters |
{ incorrect} semicolon builtin_parameters ;

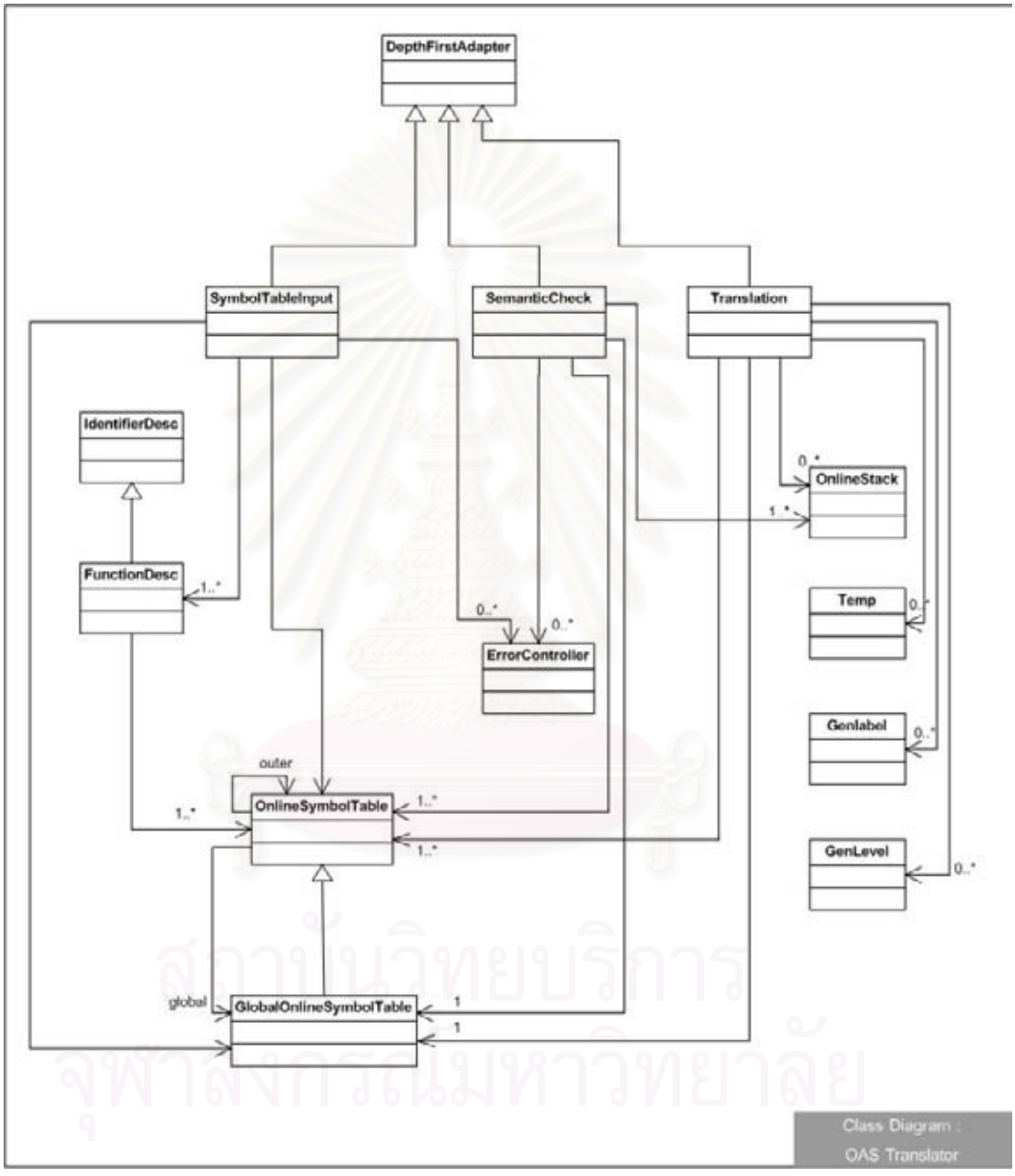
- ตัวดำเนินการเปรียบเทียบความเท่ากัน

การใช้ตัวดำเนินการเปรียบเทียบด้วยเครื่องหมาย “==” บางครั้งผู้ใช้เลือกใช้เครื่องหมาย “=” ทำให้เกิดข้อผิดพลาดขึ้นได้ ในขั้นตอนนี้ทำการปรับปรุงโดยการเพิ่มโปรดักชันทางเลือก {equal_incorrect} ในการบรรยายไวยากรณ์

equality =
{equal} equality equal relational |
{equal_incorrect} equality assign relational |
{notequal} equality notequal relational |
relational;

แผนภาพคลาสส่วนการแปลภาษาโอเอเอส

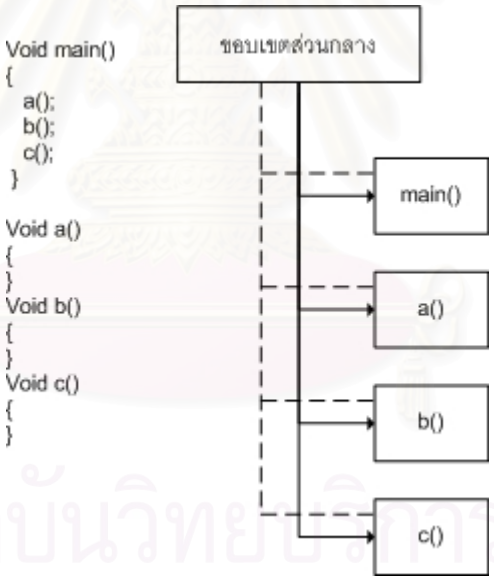
แผนภาพคลาสใช้แสดงคลาส (Class) และความสัมพันธ์ระหว่างคลาสต่างๆ เพื่อจำลองภาพการออกแบบตัวแปลภาษาโอเอเอส ความสัมพันธ์ระหว่างวัตถุ (object) ดังรูปที่ 4.7



รูปที่ 4.7 แผนภาพคลาสตัวแปลภาษา

คลาสไดอะแกรมใช้กำหนดแพคเกจเพื่อเป็นการจัดกลุ่มของคลาสที่จะทำการสร้างตัวแปลภาษา โดยมีแพคเกจที่เกี่ยวข้องดังต่อไปนี้

1. online.onlineSymbolTable เป็นแพคเกจที่รวบรวมคลาสที่ทำหน้าที่แทนตารางสัญลักษณ์ (Symbol Table) โดยจะทำการตรวจสอบข้อมูลเกี่ยวกับขอบเขตและชนิดของตัวแปร กลไลการทำงานสามารถการค้นหาตัวแปรหรือเพิ่มตัวแปรใหม่ๆได้ เนื่องจากโครงสร้างการพัฒนาตัวแปลภาษาด้วย SableCC ไม่ได้กำหนดตารางสัญลักษณ์พร้อมกับโครงสร้างเชิงวัตถุนี้ ในงานวิจัยนี้ได้สร้างคลาสที่ในแพจเกจ online.onlineSymbolTable ใช้แทนตารางสัญลักษณ์เพื่อใช้จัดเก็บข้อมูลต่างๆ ของตัวแปรที่มีอยู่ในโปรแกรมต้นฉบับ โดยตารางสัญลักษณ์ถูกออกแบบให้ทำงานเป็บแบบขอบเขตคงที่ (Static Scope) ดังรูปที่ 4.8 ฟังก์ชันและตัวแปรที่ประกาศในขอบเขตส่วนกลางสามารถอ้างอิงได้ทุกขอบเขตภายในโปรแกรมต้นฉบับ ถ้าตัวแปรในขอบเขตเฉพาะที่มีชื่อเดียวกับที่ประกาศในขอบเขตส่วนกลางจะใช้การอ้างอิงในขอบเขตเฉพาะที่เป็นหลัก โครงสร้างข้อมูลของขอบเขตจะถูกเก็บและเรียกใช้ด้วยผ่านทางตารางแฮช (Hash Tables)



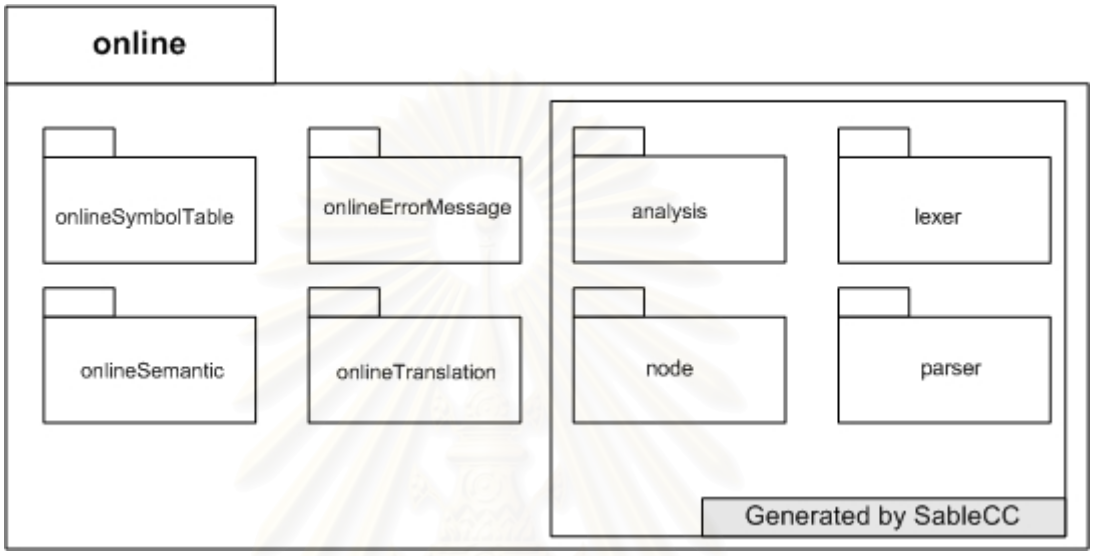
รูปที่ 4.8 ขอบเขตคงที่

2. online.onlineSemantic เป็นแพคเกจที่รวบรวมคลาสที่ทำหน้าที่ส่วนตรวจสอบความหมายของคำสั่งที่ตรวจพบในโปรแกรมต้นฉบับ

3. online.onlineTranslation เป็นแพคเกจที่รวบรวมคลาสที่ทำหน้าที่แปลงโปรแกรมต้นฉบับเป็นชุดคำสั่งอินเตอร์มิเดียท

4. online.onlineErrorMessage เป็นแพคเกจที่ทำหน้าที่ตรวจสอบข้อผิดพลาดในระหว่างการแปลภาษา

จากที่ได้กล่าวมาสามารถแสดงแพคเกจไดอะแกรมตัวแปลภาษาโอเอเอส ดังรูปที่ 4.9



รูปที่ 4.9 แพคเกจไดอะแกรมตัวแปลภาษา

รายละเอียดแพคเกจ Online.onlineSymbolTable ทำหน้าที่ใช้แทนส่วนการจัดการตารางสัญลักษณ์ ดังรูปที่ 4.9 คลาสที่เกี่ยวข้องมีดังต่อไปนี้

คลาส OnlineSymbolTable คือ คลาสใช้แสดงขอบเขตการทำงานท้องถิ่นของแต่ละฟังก์ชัน ข้อมูลตัวแปรถูกเก็บโดยใช้ตารางแฮช ชื่อตัวแปรจะประกาศได้ครั้งเดียวภายในขอบเขตแต่ละฟังก์ชัน เนื่องจากภาษาไม่สามารถประกาศฟังก์ชันแบบซ้อนในได้ (Nested Functions)

คลาส GlobalOnlineSymbolTable คือ คลาสสืบทอดคุณสมบัติจากคลาส OnlineSymbolTable ใช้แสดงขอบเขตทั่วไปของโปรแกรมต้นฉบับ และทำให้สามารถเก็บตัวแปรที่กำหนดในส่วนของตัวแปรทั่วไปได้ การอ้างอิงอินสแตนซ์ของคลาส IdentifierDesc และคลาส FunctionDesc ถูกเก็บอยู่ในตารางแฮช มีลักษณะเช่นเดียวกับคลาส OnlineSymbolTable

คลาส IdentifierDesc คือ คลาสสำหรับเก็บข้อมูลของตัวแปร เช่น ตัวระบุชื่อตัวแปร, ชนิดตัวแปร, ประเภทของตัวแปรและค่าของตัวแปร

คลาส FunctionDesc คือ คลาสสืบทอดคุณสมบัติจากคลาส IdentifierDesc ทำหน้าที่เก็บรายละเอียดฟังก์ชันนอกจากคุณสมบัติที่ได้จากคลาส IdentifierDesc นอกจากนั้นยังทำ

หน้าที่เก็บข้อมูลการส่งค่ากลับ, รายการของอาร์กิวเมนต์ และมีความสัมพันธ์กับคลาส OnlineSymbolTable เพื่อสร้างขอบเขตการทำงานของตัวแปร รายการของอาร์กิวเมนต์สร้างโดยใช้ ArrayList จัดเก็บข้อมูลเรียงตามลำดับตามการประกาศตัวแปร อินสแตนซ์ของคลาส OnlineSymbolTable ใช้สำหรับการอ้างอิงขอบเขตการทำงาน

คลาส SymbolTableInput คือคลาสสืบทอดคุณสมบัติจากคลาส DepthFirstAdapter ทำหน้าที่ท่องโหนดไวยากรณ์ต้นไม้เพื่อจัดเก็บข้อมูลตัวแปร ข้อมูลฟังก์ชัน และกำหนดขอบเขตของตัวแปรภายในโปรแกรมต้นฉบับโดยใช้คลาส OnlineSymbolTable หรือ GlobalOnlineSymbolTable ขึ้นอยู่กับขอบเขตการประกาศตัวแปร

การออกแบบตารางสัญลักษณ์นี้สามารถใช้ชื่อตัวแปรเดียวกันในขอบเขตการทำงานต่างกันได้ การสร้างตารางสัญลักษณ์จะทำหลังจากที่ส่วนตรวจสอบไวยากรณ์ได้สร้างไวยากรณ์ต้นไม้เสร็จเรียบร้อยแล้ว โดยจะทำการสร้างก่อนขั้นตอนการตรวจสอบความหมายเพราะเป็นไปได้ว่าอาจมีการอ้างอิงตัวแปร โดยที่ไม่ได้ประกาศไว้ก่อนในโปรแกรมต้นฉบับซึ่งในส่วนนี้ส่วนตรวจสอบไวยากรณ์ไม่สามารถตรวจสอบได้

ส่วนการตรวจสอบความหมายแพคเกจ online.onlineSemantic ดังรูปที่ 4.9 ประกอบด้วย คลาสที่เกี่ยวข้องดังต่อไปนี้

คลาส OnlineStack คือ คลาสสำหรับเก็บอ็อบเจกต์ข้อมูลชั่วคราว

คลาส SemanticCheck คือ คลาสสืบทอดคุณสมบัติจากคลาส DepthFirstAdapter ในส่วนนี้จะทำการเขียนแอคชันโค้ดภายในโอเวอร์ไรด์เมทธอด ตรวจสอบการเข้าและออกจากโหนดคลาสของไวยากรณ์ต้นไม้ เพื่อใช้ทำหน้าที่ตรวจสอบความหมายของประโยคคำสั่ง โดยใช้คลาส OnlineStack ช่วยในการจัดเก็บข้อมูลและเรียกใช้ข้อมูลนำมาเปรียบเทียบ โดยจะตรวจสอบว่าตัวแปรต้องมีชนิดเดียวกันหรือเข้ากันได้ (Compatible)

ส่วนการแปลงคำสั่งอินเตอร์มิเดียทอยู่ในแพจแกจ online.onlineTranslation ดังรูปที่ 4.9 โดยแสดงคลาสที่เกี่ยวข้อง ดังต่อไปนี้

คลาส Translation คือ คลาสสำหรับทำหน้าที่แปลงโปรแกรมต้นฉบับให้อยู่ในรูปแบบคำสั่งอินเตอร์มิเดียท สืบทอดคุณสมบัติจากคลาส DepthFirstAdapter ในส่วนนี้จะทำการเขียนแอคชันโค้ดลงในโอเวอร์ไรด์เมทธอด ตรวจสอบการเข้าและออกจากโหนดคลาสไวยากรณ์ต้นไม้เพื่อใช้แปลงเป็นคำสั่งอินเตอร์มิเดียทและจัดเก็บลงแฟ้มข้อมูลผลลัพธ์ (Output File)

คลาส Temp คือ คลาสสำหรับสร้างตัวเก็บข้อมูลชั่วคราว จากตัวกระทำ เช่น ตัวกระทำทางคณิตศาสตร์ หรือตัวกระทำตรรกะ เป็นต้น ใช้ในขั้นตอนการแปลงคำสั่งอินเตอร์มิเดียท

คลาส Genlabel คือ คลาสสำหรับใช้สร้างสัญลักษณ์เลเบลในคำสั่งการควบคุมลำดับการประมวลผล ใช้ในขั้นตอนการแปลงคำสั่งอินเตอร์มิเดียท

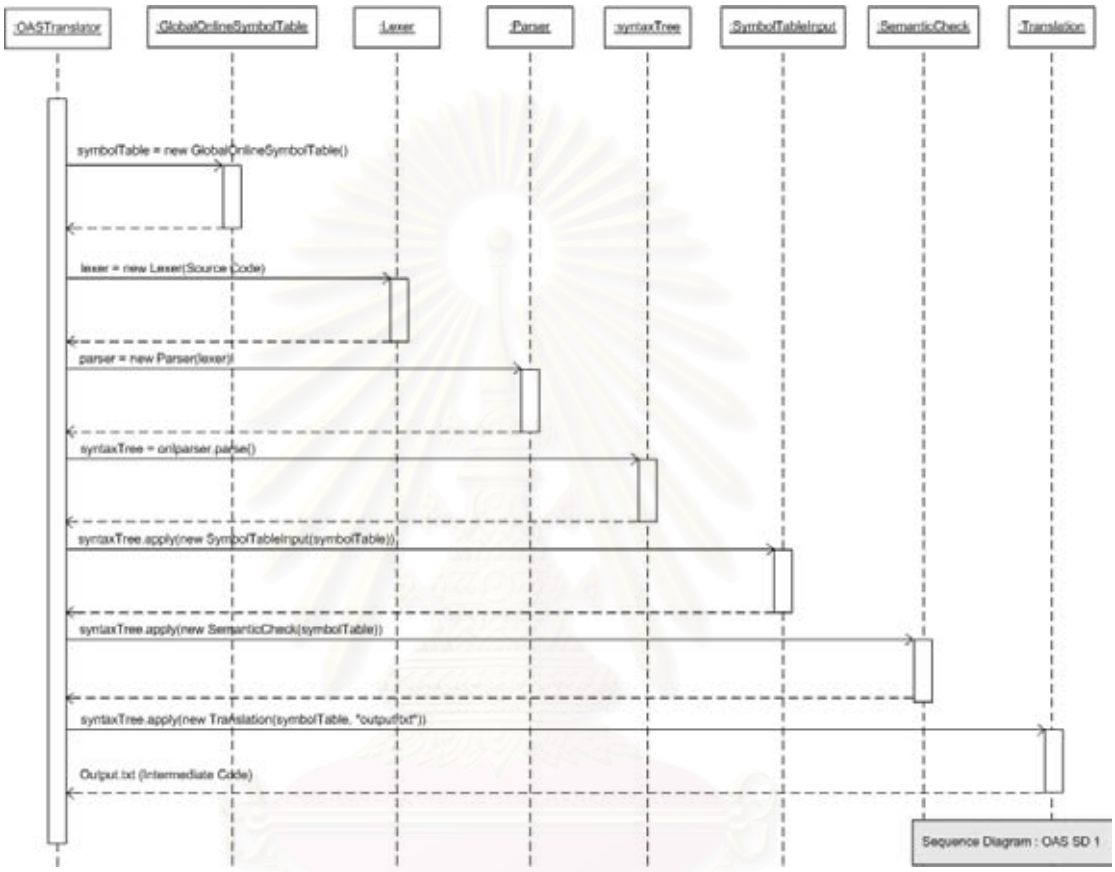
คลาส GenLevel คือ คลาสสำหรับสร้างสัญลักษณ์อ้างอิงระดับของตัวแปรใช้ในขั้นตอนการแปลงคำสั่งอินเตอร์มิเดียทเพื่อช่วยแสดงภาพความสัมพันธ์ของตัวแปร

ส่วนการตรวจสอบข้อผิดพลาดจัดเก็บอยู่ในแพคเกจ online.onlineErrorMessage แสดง ดังรูปที่ 4.12 จากภาพจะแสดงให้เห็นคลาสที่เกี่ยวข้อง ดังนี้

คลาส ErrorController คือ คลาสที่เก็บรายละเอียดเกี่ยวกับข้อผิดพลาดในระหว่างการตรวจสอบโปรแกรมต้นฉบับ มีความสัมพันธ์กับคลาส SymbolTableInput ถ้าตรวจพบความผิดพลาดในระหว่างการนำข้อมูลจัดเก็บในตารางสัญลักษณ์ และสัมพันธ์กับคลาส SemanticCheck กรณีตรวจสอบชนิดข้อมูลและการตรวจจับข้อผิดพลาดโปรดักชัน

แผนภาพซีเควนซ์ส่วนการแปลภาษาโอเอเอส

แผนภาพซีเควนซ์แสดงความสัมพันธ์ในลักษณะการโต้ตอบระหว่างวัตถุในระบบ และการส่งข้อความ (Message) การทำงานของตัวแปลภาษาโอเอเอสมีลำดับของกิจกรรม ดังรูปที่ 4.10



รูปที่ 4.10 ภาพรวมกิจกรรมตัวแปลภาษา

จากแผนภาพซีเควนซ์รูปที่ 4.10 คือ ภาพรวมกิจกรรมตัวแปลภาษาสามารถอธิบายรายละเอียดลำดับกิจกรรมส่วนการแปลภาษา ดังนี้

1. OasTranslator คือ คลาสหลักสำหรับใช้ควบคุมการทำงานของส่วนแปลภาษา โดยจะสร้างอ็อบเจกต์ symbolTable จาก GlobalOnlineSymbolTable ทำหน้าที่เก็บขอบเขตการอ้างอิงตัวแปรส่วนกลางและส่งการอ้างอิงค่าให้การทำงานในส่วนอื่นๆ คือ การจัดเก็บข้อมูลในตารางสัญลักษณ์, การตรวจสอบชนิดตัวแปร และการสร้างคำสั่งอินเตอร์มิเดียท

2. OasTranslator สร้างส่วนการวิเคราะห์คำ คือ อ็อบเจกต์ lexer โดยใช้โปรแกรมต้นฉบับเป็นพารามิเตอร์

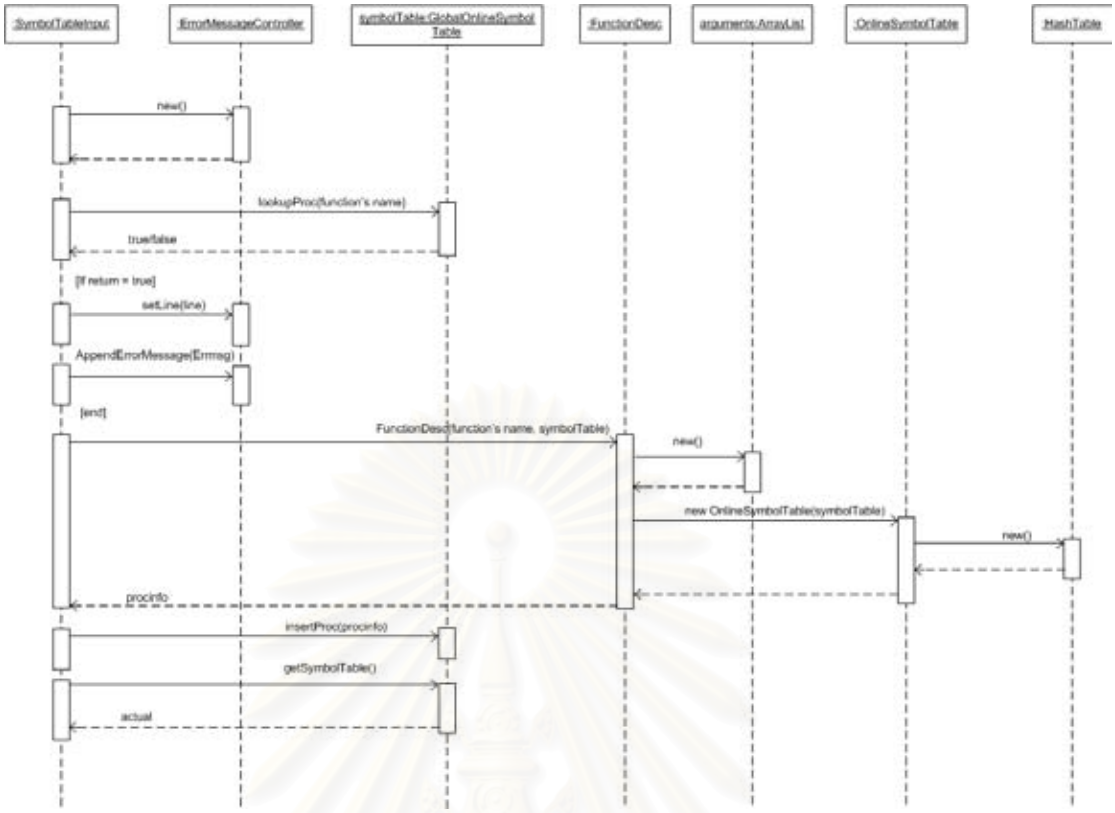
3. OasTranslator สร้างส่วนการวิเคราะห์รูปประโยค คือ อ็อบเจกต์ parser โดยใช้อ็อบเจกต์ lexer เป็นพารามิเตอร์

4. OasTranslator เรียกใช้เมทธอด parse () ของอ็อบเจกต์ parser เพื่อใช้ตรวจสอบโปรแกรมต้นฉบับตามที่กำหนดไว้ในไวยากรณ์ภาษาและคืนค่าเป็นอ็อบเจกต์ syntaxTree

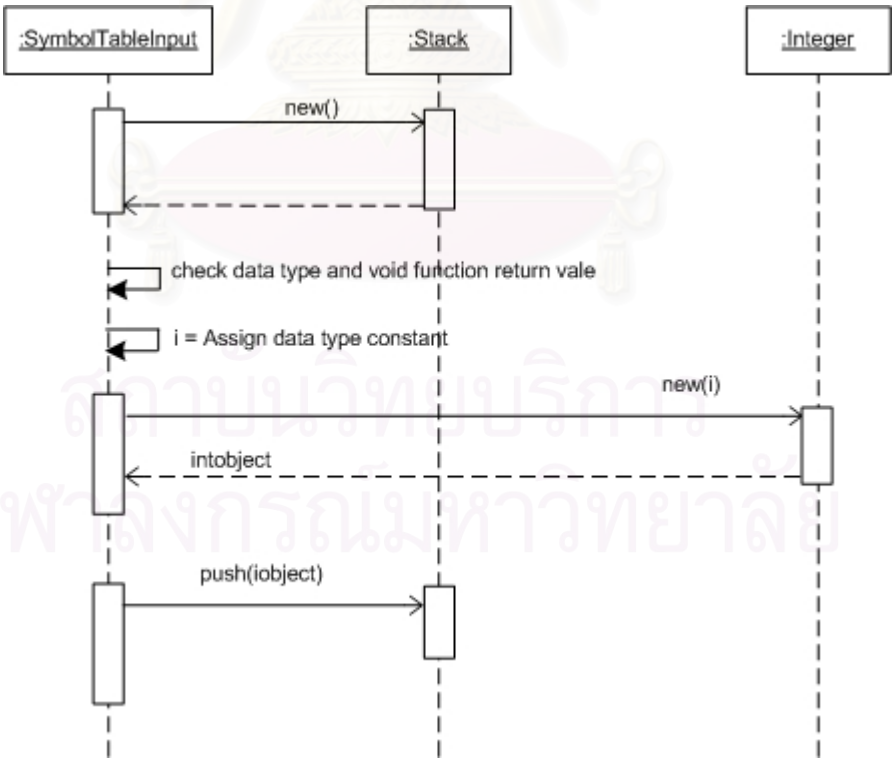
5. ขั้นตอนการจัดเก็บข้อมูลลงในตารางสัญลักษณ์เกิดขึ้นหลังจากส่วนการวิเคราะห์และส่วนการวิเคราะห์รูปประโยคทำงานถูกต้องแล้ว OASTranslator จะสร้างอ็อบเจกต์ SymbolTableInput จากนั้นส่งผ่านการเรียกด้วยเมทธอด apply () เรียกใช้โอเวอร์โรดเมทธอดที่ถูกเขียนอยู่ในคลาส SymbolTableInput ทำหน้าที่กำหนดขอบเขตและจัดเก็บข้อมูลตัวแปร เช่น ชื่อ, ชนิด, ประเภทของตัวแปร เป็นต้น การทำงานจะตรวจสอบข้อผิดพลาดโดยใช้อ็อบเจกต์ ErrorController เช่น กรณีการจัดเก็บตัวแปรที่มีอยู่แล้วในตารางสัญลักษณ์ภายในขอบเขตเดียวกัน เป็นต้น

ขบวนการทำงานของการจัดเก็บข้อมูลในตารางสัญลักษณ์ มีขั้นตอนการทำงานดังนี้

1.กำหนดขอบเขตการทำงานของตัวแปร เริ่มจากการกำหนดขอบเขตส่วนกลางเป็นขอบเขตเริ่มต้น เมื่อพบการประกาศฟังก์ชันจะทำการค้นหาฟังก์ชันในขอบเขตส่วนกลาง ผ่านเมทธอด lookupProc () ถ้าพบการประกาศฟังก์ชันซ้ำซ้อนจะส่งข้อผิดพลาดจัดเก็บในอ็อบเจกต์ Errormsg ถ้าพบว่าเป็นฟังก์ชันใหม่จะทำการสร้างอ็อบเจกต์ FunctionDesc ใช้แทนการจัดเก็บตัวแปรในส่วนท้องถิ่นโดยผ่านค่าพารามิเตอร์สำหรับอ้างอิงค่าจากขอบเขตส่วนกลาง ทำให้สามารถอ้างอิงค่าจากขอบเขตส่วนกลางได้ถ้าพบว่าตัวแปรที่ค้นหาไม่ได้ประกาศในขอบเขตท้องถิ่นของฟังก์ชันนั้นๆ ขอบเขตส่วนท้องถิ่นจะจัดเก็บพารามิเตอร์โดยใช้อ็อบเจกต์ ArrayList และจัดเก็บหรือค้นหาตัวแปรผ่านทางอ็อบเจกต์ Hashtable โดยที่อ็อบเจกต์ FunctionDesc เมื่อสร้างเสร็จแล้วจะถูกจัดเก็บผ่านทางเมทธอด insertProc () ไว้ในขอบเขตส่วนกลาง ภายหลังจากที่จัดเก็บแล้วการอ้างอิงขอบเขตปัจจุบันจะถูกเรียกใช้ผ่านเมทธอด getSymbolTable () สำหรับใช้ในการจัดเก็บและเรียกใช้งานตัวแปรต่อไป ดังรูปที่ 4.11



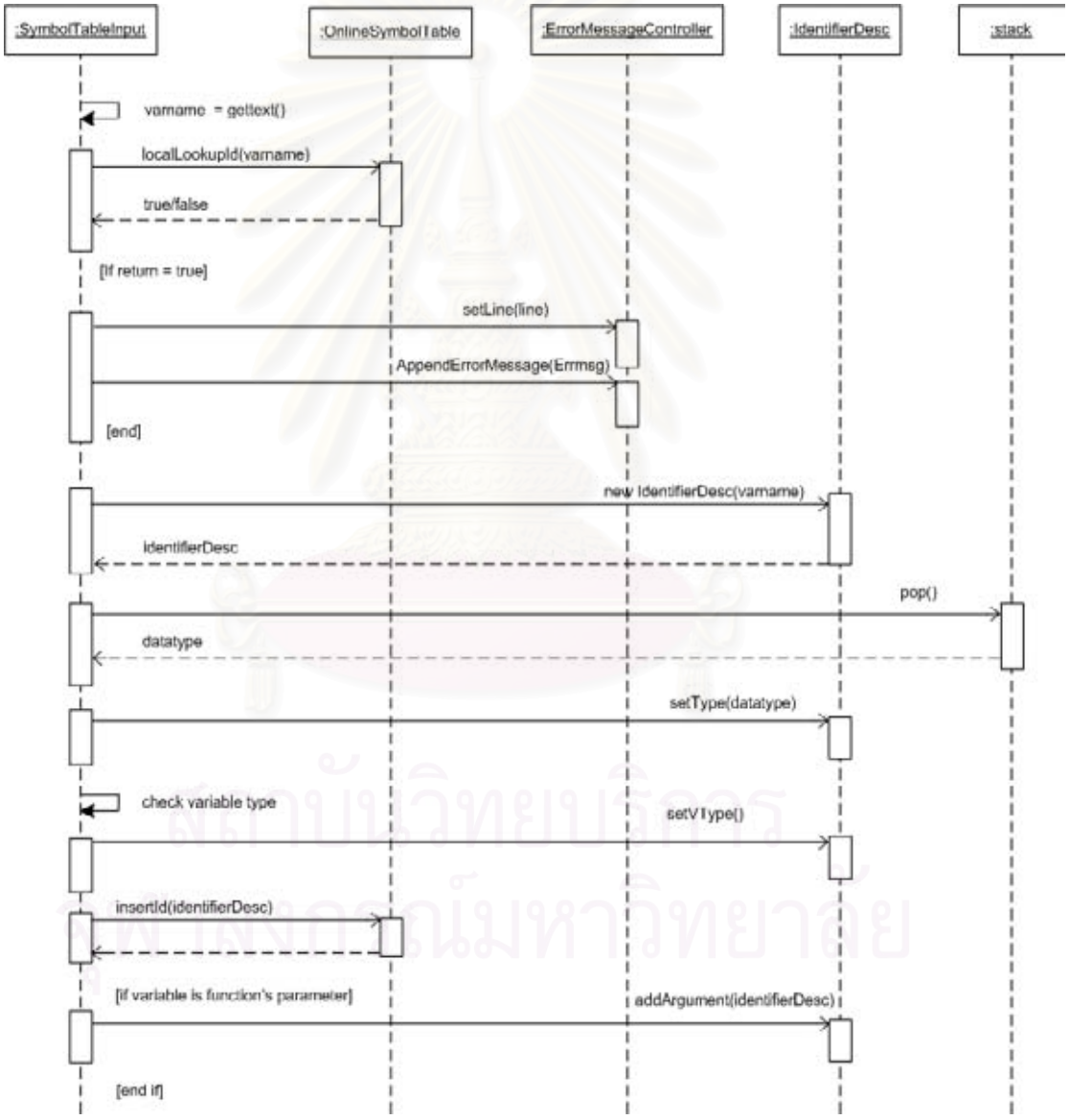
รูปที่ 4.11 แผนภาพซีเควนซ์การสร้างขอบเขตการทำงานของตัวแปร



รูปที่ 4.12 แผนภาพซีเควนซ์การกำหนดชนิดข้อมูล

2.การกำหนดชนิดข้อมูล เป็นขั้นตอนการตรวจสอบชนิดข้อมูลและชนิดฟังก์ชันแบบไม่ส่งค่ากลับ จะกำหนดด้วยค่าคงที่แยกชนิดข้อมูลแต่ละชนิดและจัดเก็บไว้ใน อ็อบเจกต์ Stack ผ่านเมทธอด push () โดยจะถูกเรียกใช้ในขั้นตอนการการจัดเก็บตัวแปรต่อไป ดังรูปที่ 4.12

3.การจัดเก็บตัวแปร ตรวจสอบจากการประกาศตัวแปรนำเข้า, ตัวแปรผลลัพธ์, ตัวแปรทั่วไป, ตัวแปรแบบแถวลำดับ และพารามิเตอร์ของฟังก์ชัน โดยค้นหาชื่อตัวแปรในตารางสัญลักษณ์ ผ่านเมทธอด localLookupId ()



รูปที่ 4.13 แผนภาพซีเควนซ์การจัดเก็บตัวแปร

ถ้าพบจะจัดเก็บข้อผิดพลาดไว้แสดงผลให้ผู้ใช้งานทราบผ่านทางอ็อบเจกต์ ErrorMessageController ถ้าไม่พบตัวแปรที่ประกาศไว้ก็จะทำการสร้างอ็อบเจกต์ IdentifierDesc และกำหนดชนิดข้อมูลตัวแปรคือค่าคงที่จากอ็อบเจกต์ OnlineStack ที่ได้จากขั้นตอนการตรวจสอบชนิดตัวแปร ผ่านเมทธอด setType () จากนั้นตรวจสอบและกำหนดประเภทของตัวแปรว่าเป็นตัวแปรชนิดทั่วไป หรือเป็นชนิดแถวลำดับ ผ่านเมทธอด setVType () เมื่อได้ข้อมูลของตัวแปรแล้วจะจัดเก็บข้อมูลไว้ในตารางสัญลักษณ์ผ่านเมทธอด insertId () ถ้าการตรวจสอบพบว่าตัวแปรที่ประกาศนั้น เป็นพารามิเตอร์ของฟังก์ชันจะจัดเก็บข้อมูลของตัวแปรเป็นอาร์คิวเมนต์ไว้ในอ็อบเจกต์ FunctionDesc ดังรูปที่ 4.13

6. ขั้นตอนการตรวจสอบความหมาย OASTranslator สร้างอ็อบเจกต์ SemanticCheck ผ่านการเรียกใช้เมทธอด apply () ซึ่งจะเรียกใช้โอเวอร์ไรด์เมทธอดที่ถูกเขียนอยู่ในคลาส SemanticCheck โดยใช้อ็อบเจกต์ symbolTable เป็นพารามิเตอร์สำหรับใช้อ้างอิงขอบเขตการทำงานและค้นหาชนิดข้อมูลของตัวแปรในตารางสัญลักษณ์ การทำงานจะใช้อ็อบเจกต์ OnlineStack จัดเก็บชนิดข้อมูลชั่วคราวใช้ในการเปรียบเทียบ การตรวจสอบความหมายจะทำการตรวจสอบคำสั่ง ดังนี้

คำสั่งการกำหนดค่า เป็นการหาความสอดคล้องของชนิดข้อมูลทางซ้ายของเครื่องหมาย "=" จะต้องเป็นข้อมูลชนิดเดียวกันทางขวา ข้อมูลด้านซ้ายมือ ประกอบด้วยตัวแปรซึ่งแบ่งออกเป็นสองประเภทคือตัวแปรทั่วไป และตัวแปรแบบแถวลำดับ การค้นหาตัวแปรจะเรียกใช้จากตารางสัญลักษณ์ ถ้าพบจะเรียกใช้ตัวแปรโดยนำชนิดและประเภทของตัวแปรขึ้นมาตรวจสอบ จากนั้นจัดเก็บชนิดตัวแปรไว้ในอ็อบเจกต์ OnlineStack รอการตรวจสอบกับชนิดข้อมูลด้านขวาของเครื่องหมาย "=“ ด้านขวาของเครื่องหมาย “=” ค่าที่นำมาตรวจสอบความหมาย มีดังนี้

ค่าคงที่และฟังก์ชันภายใน ถ้าตรวจพบค่าคงที่หรือฟังก์ชันภายใน ที่ถูกกำหนดค่าให้กับตัวแปรด้านซ้ายมือ จะตรวจสอบชนิดของค่าคงที่นั้น และจัดเก็บชนิดของข้อมูลไว้ใน OnlineStack

เมื่อออกจากโหนดการกำหนดค่าหรือโหนด "=" จะนำชนิดข้อมูลที่จัดเก็บอยู่ใน OnlineStack ขึ้นมาเปรียบเทียบความสอดคล้องของชนิดข้อมูลนั้นๆ

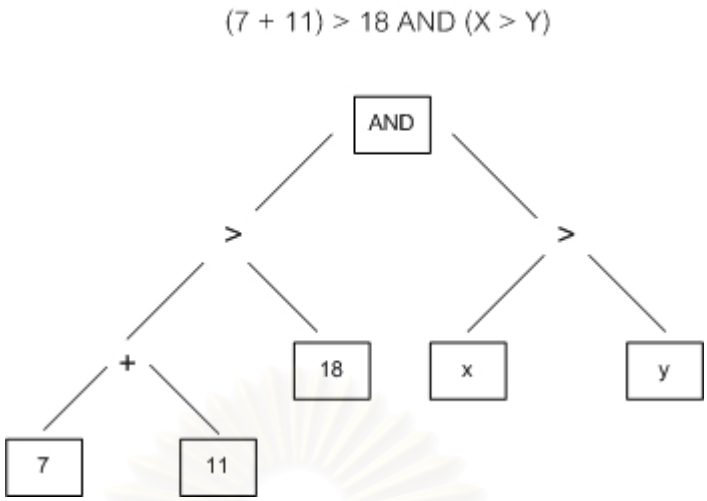
การเรียกใช้ฟังก์ชัน เมื่อพบการเรียกใช้ฟังก์ชันจะทำการตรวจสอบว่าในโปรแกรมได้มีการประกาศฟังก์ชันที่เรียกใช้หรือไม่ ถ้าพบว่ามีการประกาศ จะตรวจสอบพารามิเตอร์ทั้งหมดของฟังก์ชันโดยจะตรวจสอบชนิด และจำนวนพารามิเตอร์ว่าสอดคล้องกับจำนวนพารามิเตอร์ที่ได้ประกาศไว้หรือไม่จากตารางสัญลักษณ์ เมื่อออกจากการท่องโหนดในแต่ละพารามิเตอร์ของฟังก์ชัน

นิพจน์ เป็นการหาผลลัพธ์ของตัวถูกดำเนินการอย่างน้อย 2 จำนวนด้วยเครื่องหมายใดๆ ที่กำหนดให้โดยที่ชนิดของตัวดำเนินการทั้งสองต้องมีชนิดเดียวกัน เมื่อท่องโหนดผ่านตัวดำเนินการตัวที่หนึ่งและสองจะจัดเก็บชนิดของตัวดำเนินการไว้ใน OnlineStack หลังจากท่องออกจากโหนดตัวดำเนินการใดๆแล้ว จะนำค่าที่เก็บอยู่ใน OnlineStack ทั้งสองออกมาเปรียบเทียบความสอดคล้องกัน ถ้าพบว่าถูกต้องก็จะนำชนิดที่ตรวจสอบนี้จัดเก็บลงใน OnlineStack รอการเปรียบเทียบจากค่าที่อยู่ทางซ้ายของเครื่องหมาย "=" ต่อไป

คำสั่งเลือกและวนซ้ำ การตรวจสอบคำสั่งทางเลือกและการวนซ้ำ คำสั่งทางเสือกใช้คำหลัก "if" ในขณะที่คำสั่งวนซ้ำใช้คำหลัก "while" ในการกำหนดคำสั่ง ทั้งสองคำสั่งใช้หลักการตรวจสอบความหมายคล้ายกัน โดยตรวจสอบจากเงื่อนไขสำหรับการตัดสินใจในการเลือกทำงาน การตรวจสอบจะตรวจสอบว่าค่านิพจน์ของเงื่อนไขจะต้องเป็นชนิดบลูลีนเสมอ

การส่งค่ากลับ เมื่อใดที่มีการท่องโหนดฟังก์ชันใดๆ จะตรวจสอบว่าฟังก์ชันนั้นมีประโยคคำสั่งการส่งค่ากลับหรือไม่ จากการนิยามไวยากรณ์รูปแบบฟังก์ชันแยกออกเป็นสองรูปแบบ ทำให้สามารถตรวจสอบความถูกต้องได้ถ้าพบว่าโดยจะแสดงข้อผิดพลาด ถ้าฟังก์ชันประกาศไว้แบบไม่มีการส่งค่ากลับ หรือ Void แต่พบว่ามีประโยคการส่งค่ากลับภายในฟังก์ชันที่ประกาศไว้

จากตัวอย่างรูปที่ 4.14 การทำงานของการตรวจสอบนิพจน์บูลลีน (7 + 11) > 18 AND (X > Y) การทำงานส่วนการตรวจสอบความหมายมีขั้นตอนการทำงานตรวจสอบความหมายของชนิดข้อมูล โดยการท่องโหนดไวยากรณ์ต้นไม้ มีขั้นตอนดังนี้



รูปที่ 4.14 ไวยากรณ์ต้นไม้ของนิพจน์ (7 + 11) > 18 AND (X > Y)

สถานะกองซ้อนที่ 1 ขณะท่องโหนดคลาส Semantic ตรวจสอบพบค่าของ “7” หลังจากออกจากโหนดแม่จะทำการตรวจสอบชนิดข้อมูล เมื่อพบชนิดข้อมูลเป็นจำนวนเต็ม จากนั้นจะทำการเก็บค่าอินสแตนซ์จำนวนเต็มไว้ในกองซ้อน จากนั้นคลาส tree-walker ท่องไปในโหนดพบค่า “11” ทำการเก็บค่าอินสแตนซ์จำนวนเต็มไว้ในกองซ้อน และกลับสู่โหนดแม่

ตารางที่ 4.2 สถานะกองซ้อนของการตรวจสอบความหมาย

สถานะกองซ้อน	ชนิดข้อมูลในกองซ้อน	ค่าในโหนดไวยากรณ์ต้นไม้
1	INTEGER	7
	INTEGER	11
2	INTEGER	+
3	INTEGER	18
	INTEGER	+
4	BOOLEAN	>
5	INTEGER	Y
	INTEGER	X
	BOOLEAN	>
6	BOOLEAN	>
	BOOLEAN	>
7	BOOLEAN	AND

สถานะกองซ้อนที่ 2 เมื่อ คลาส tree-walker ออกจากโหนด “+” ไปยังโหนดอินสแตนซ์จำนวนเต็มทั้งสองค่า จะถูกดึงขึ้นมาจากกองซ้อนและตรวจสอบว่าใช้กับตัวดำเนินการ “+” ได้หรือไม่ ถ้าได้ก็จะทำการเก็บค่าอินสแตนซ์จำนวนเต็มกลับลงไปในกองซ้อน

สถานะกองซ้อนที่ 3 จากนั้นคลาส tree-walker ท่องเข้าไปพบค่า “8” ทำการเก็บค่าอินสแตนซ์จำนวนเต็มไว้ในกองซ้อน

สถานะกองซ้อนที่ 4 คลาส tree-walker ท่องออกจากโหนด “>” อินสแตนซ์จำนวนเต็มทั้ง 2 ค่าที่อยู่ในกองซ้อนจะถูกดึงขึ้นมาจากกองซ้อนเพื่อตรวจสอบชนิดข้อมูลจากนั้นทำการเก็บค่าบูลลีนลงกองซ้อนเนื่องจากใช้เครื่องหมายเปรียบเทียบ “>” เป็นตัวดำเนินการ

สถานะกองซ้อนที่ 5 เมื่อท่องเข้าไปพบตัวแปร “X” จะทำการค้นหาชนิดข้อมูลของตัวแปร “X” ในตารางสัญลักษณ์และเก็บค่าอินสแตนซ์ชนิดข้อมูลในที่นี้คือจำนวนเต็มลงในกองซ้อน ในกรณีนี้ตัวแปร “Y” ชนิดข้อมูลคือตัวเดียวกันกับตัวแปร “X” คือจำนวนเต็ม ทำการเก็บค่าอินสแตนซ์ชนิดข้อมูลของตัวแปร “Y” ลงในกองซ้อน

สถานะกองซ้อนที่ 6 ออกจากโหนด “>” จะทำการดึงอินสแตนซ์ 2 จำนวนขึ้นมาเปรียบเทียบ ตัวดำเนินการ “>“ จะให้ค่าผลลัพธ์เป็นข้อมูลชนิดบลูลีนจากนั้นเก็บค่าอินสแตนซ์ชนิดข้อมูลบลูลีนลงในกองซ้อน

สถานะกองซ้อนที่ 7 ออกจากโหนด “AND” จะทำการตรวจสอบ 2 อินสแตนซ์ที่เหลืออยู่ในกองซ้อน ผลลัพธ์ที่ได้คือข้อมูลชนิดบลูลีน

7. ขั้นตอนการแปลงคำสั่งอินเตอร์มิเดียท เป็นขั้นตอนแปลงโปรแกมต้นฉบับให้อยู่ในรูปแบบคำสั่งอินเตอร์มิเดียทจัดเก็บไว้ในแฟ้มข้อมูลผลลัพธ์ โดยที่รูปแบบทรีแอดเดรสโค้ด คือ ลำดับของคำสั่งในรูปแบบ ดังนี้

X = Y op Z (คำสั่งมีตัวกระทำเพียงตัวเดียว)

โดยที่ X, Y, Z อาจจะเป็นชื่อ ค่าคงที่ หรือตัวเก็บข้อมูลชั่วคราว ส่วน “op” คือตัวกระทำ เช่น ตัวกระทำทางคณิตศาสตร์ ตัวกระทำตรรกะ หรือตัวกระทำระดับบิต

ตัวอย่างเช่น X + Y * Z จะถูกแปลงให้อยู่ในรูปแบบทรีแอดเดรสโค้ด ได้ดังนี้

temp1 = Y * Z

temp2 = X + temp1

เมื่อ temp1 และ temp2 คือตัวเก็บข้อมูลชั่วคราวที่โปรแกรมตัวแปลภาษาเป็นผู้สร้างขึ้น งานวิจัยนี้ใช้คำสั่งทรีแอดเดรสโค้ด ประกอบด้วยสัญลักษณ์ต่างๆ และคำสั่งควบคุมการทำงาน ดังนี้

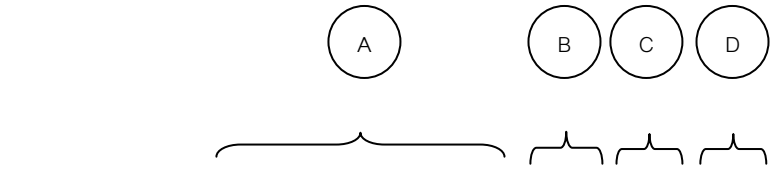
- ประโยคกำหนดค่า (Assignment Statements) คือ X = Y op Z โดยที่ op คือคำสั่งทางคณิตศาสตร์ หรือคำสั่งทางตรรกะ เป็นต้น
- คำสั่งกำหนดค่า (Assignment Instructions) คือ X = op Y โดยที่ op คือตัวกระทำแบบมีเครื่องหมาย
- ประโยคสร้างสำเนา (Copy Statement) คือ X = Y
- คำสั่งกระโดดข้ามอย่างมีเงื่อนไข (Conditional Jump) คือ if X relop Y goto L โดยที่ relop คือ ตัวดำเนินการการเปรียบเทียบ goto L คือ คำสั่งกระโดดข้าม L คือเลเบลใช้เครื่องหมาย '@' ตามด้วยตัวเลขที่ไม่ซ้ำกันและเครื่องหมาย ':' เช่น goto @1:, goto @2: เป็นต้น
- การเรียกใช้ฟังก์ชัน (Function Calls) มีรูปแบบเมื่อ param xi คือ ค่าตัวแปรที่ส่งไปยังฟังก์ชันที่เรียก call p, n คือการเรียกใช้ฟังก์ชัน p พร้อมพารามิเตอร์จำนวน n ตัว ดังนี้

```
param x1
  ....
param xn
Call p, n
```

โดยมีความหมายการเรียกใช้เหมือนคำสั่ง p(x1, x2…xn)

- การกำหนดดรรชนีย์ (Indexed Assignment) อยู่ในรูปแบบ X = Y[i] หรือ X[i] = Y
- ตัวเก็บข้อมูลชั่วคราวใช้สัญลักษณ์ '$t' ตามด้วยตัวเลขที่ไม่ซ้ำกัน เช่น $t1, $t2 เป็นต้น

คำสั่งอินเตอร์มิเดียทใช้รูปแบบทรีแอดเดรสโค้ด สำหรับนำไปใช้สร้างภาพมโนทัศน์อัลกอริทึมมีรูปแบบ ดังนี้



โดยที่

A คือ ส่วนคำสั่งอินเตอร์มิเดียท

B คือ ส่วนเลขแสดงบรรทัดเริ่มต้นของโค้ดโปรแกรมต้นฉบับที่ถูกแปลงเป็นคำสั่งอินเตอร์มิเดียทสำหรับใช้ในการทำแถบแสงแสดงความสัมพันธ์ของรูปภาพกับคำสั่งการทำงาน

C คือ ส่วนเลขแสดงบรรทัดสุดท้ายของโค้ดโปรแกรมต้นฉบับที่ถูกแปลงเป็นคำสั่งอินเตอร์มิเดียทสำหรับใช้ในการทำแถบแสงแสดงความสัมพันธ์ของรูปภาพกับคำสั่งการทำงาน

D คือ ส่วนระดับของคำสั่งเป็นตัวบอกอยู่

ตารางที่ 4.3 อธิบายรูปแบบการแปลงคำสั่งโปรแกรมต้นฉบับเป็นคำสั่งอินเตอร์มิเดียท

คำสั่งโปรแกรมต้นฉบับ	หน้าที่	คำอธิบายรูปแบบคำสั่งอินเตอร์มิเดียท	ตัวอย่าง
input x ;	ประกาศตัวแปรนำเข้า	input varname โดยที่ varname คือ ชื่อตัวแปรนำเข้า	input x
output z ;	ประกาศตัวแปรผลลัพธ์	output varname โดยที่ varname คือ ชื่อตัวแปรผลลัพธ์	output z
int a;	ประกาศตัวแปร	dim scope_type_varname โดยที่ scope คือ ขอบเขตที่ประกาศตัวแปร เช่น ประกาศในส่วนการประกาศตัวแปรทั่วไป scope จะมีค่าเท่ากับ global หากประกาศในส่วนภายในของฟังก์ชันใดๆ scope จะเป็นชื่อของฟังก์ชันนั้นแทน type คือ ชนิดตัวแปร varname คือ ชื่อตัวแปร	global_integer_a หรือ main_integer_a

ตารางที่ 4.3 (ก) อธิบายรูปแบบการแปลงคำสั่งโปรแกรมต้นฉบับเป็นคำสั่งอินเตอร์มิเดียท

คำสั่งโปรแกรมต้นฉบับ	หน้าที่	คำอธิบายรูปแบบคำสั่งอินเตอร์มิเดียท	ตัวอย่าง
int a[2];	ประกาศตัวแปรแบบแถวลำดับ	dim scope_type_varname#1 dim scope_type_varname#2 รูปแบบการประกาศเหมือนกับในส่วนการประกาศตัวแปรแบบไม่เป็นแบบแถวลำดับ แตกต่างที่การประกาศแบบแถวลำดับจะใช้เครื่องหมาย '#' และตามด้วยเลขลำดับ	global_integer_a#1 global_integer_a#2 หรือ main_integer_a#1 main_integer_a#2

void main()
{
}
	การประกาศฟังก์ชัน	function_name :
		โดยที่ function_name คือ ชื่อฟังก์ชันตามด้วยเครื่องหมาย ":"
			main:

ตารางที่ 4.3 (ข) อธิบายรูปแบบการแปลงคำสั่งโปรแกรมต้นฉบับเป็นคำสั่งอินเตอร์มิเดียท

คำสั่งโปรแกรมต้นฉบับ	หน้าที่	คำอธิบายรูปแบบคำสั่งอินเตอร์มิเดียท	ตัวอย่าง

while (j > 0)
{
}
	คำสั่งวนรอบ	รูปแบบใช้ @1 เป็นเลเบลเริ่มต้น
		t$1 คือ ตัวแปรชั่วคราวสำหรับใช้ตรวจสอบเงื่อนไข ถ้าเงื่อนไขถ้าเป็นจริงจะทำคำสั่ง GOTO @2: ถ้าเงื่อนไขเป็นเท็จ จะทำคำสั่ง GOTO @3 และออกจากจุดวนซ้ำ
		
		@1:
		temp = j > 0
		if temp = true GOTO @2:
		GOTO @3:
		@2:
		GOTO @1:
		@3:
			@1:
			$t1= j > 0
			if $t1 = true GOTO @2:
			GOTO @3:
			@2:
			GOTO @1:
			@3

ตารางที่ 4.3 (ค) อธิบายรูปแบบการแปลงคำสั่งโปรแกรมต้นฉบับเป็นคำสั่งอินเตอร์มิเดียท

คำสั่ง โปรแกรม ต้นฉบับ	หน้าที่	คำอธิบาย รูปแบบคำสั่งอินเตอร์มิเดียท	ตัวอย่าง

If (j > 0)
{
คำสั่ง
ถ้าเงื่อนไข
เป็นจริง…
}
	คำสั่งเงื่อนไข	รูปแบบจะสร้างตัวแปรชั่วคราว t$1 สำหรับใช้ตรวจสอบเงื่อนไข ถ้าเงื่อนไขถ้าเป็นจริง จะทำคำสั่ง บรรทัดที่ต่อจาก @1: ถ้าหากไม่เข้าเงื่อนไข จะทำคำสั่ง บรรทัดที่ต่อจาก @2:	$t1 = main_integer_j > 0;
			if $t1 = true GOTO @1;
			GOTO @2:
			@1:
			คำสั่ง
			ถ้าเงื่อนไข
			เป็นจริง…
			@2:

ตารางที่ 4.3 (ง) อธิบายรูปแบบการแปลงคำสั่งโปรแกรมต้นฉบับเป็นคำสั่งอินเตอร์มิเดียท

คำสั่งโปรแกรมต้นฉบับ	หน้าที่	คำอธิบายรูปแบบคำสั่งอินเตอร์มิเดียท	ตัวอย่าง

If (j > 0)
{
a = 20;
}
else
{ z = a;
}
		รูปแบบจะสร้างตัวแปรชั่วคราว t$1 สำหรับใช้ตรวจสอบเงื่อนไข ถ้าเงื่อนไขถ้าเป็นจริงจะทำคำสั่ง บรรทัดที่ต่อจาก @1: ถ้าหากเงื่อนไขเป็นเท็จ จะทำคำสั่ง บรรทัดที่ต่อจาก GOTO @1: จนกว่าจะพบ GOTO @2: จึงจะออกจากเงื่อนไข	$t1 = main_integer_j > 0
			if $t1 = true GOTO @1:
			global_integer_z = main_integer_a
			GOTO @2:
			@1:
			main_integer_a = 20
			@2:

Int add(int a,int b)
{
}
	การประกาศฟังก์ชัน	รูปแบบจะสร้างชื่อฟังก์ชันตามด้วย “:” บอกขอบเขตการทำงานของฟังก์ชันพร้อมทั้งกำหนดพารามิเตอร์เพื่อรับค่าตัวแปรจากภายนอกฟังก์ชันด้วย #param	add:
			dim test_integer_b
			test_integer_b = #param2
			dim test_integer_a
			test_integer_a = #param1

ตารางที่ 4.3 (จ) อธิบายรูปแบบการแปลงคำสั่งโปรแกรมต้นฉบับเป็นคำสั่งอินเตอร์มิเดียท

คำสั่งโปรแกรมต้นฉบับ	หน้าที่	คำอธิบายรูปแบบคำสั่งอินเตอร์มิเดียท	ตัวอย่าง
add(2,6)	การเรียกใช้ฟังก์ชัน	param xi param xn Call func_name, numofparam	param 2 param 6 call add,2 $t1 = #return

จากตารางที่ 4.3 คำสั่งอินเตอร์มิเดียทในรูปแบบทรีแอดเดรสโค้ด ใช้กฎการแปลความหมาย (Semantic Rules) จากตารางที่ 4.4 ฟังก์ชัน “write” ใช้เพื่อแสดงว่าได้สร้างคำสั่งรูปแบบทรีแอดเดรสโค้ด จัดเก็บไว้ในแฟ้มข้อมูลผลลัพธ์ โดยสามารถค้นหาข้อมูลตัวแปรในตารางสัญลักษณ์จากฟังก์ชัน getId (id.name) โดยชื่อตัวแปรถูกแทนที่ด้วย id และถูกเก็บไว้ในแอตทริบิวต์ id.name

ตารางที่ 4.4 การใช้กฎการแปลความหมายสร้างรูปแบบคำสั่งทรีแอดเดรสโค้ด

โปรดักชัน	กฎการแปลความหมาย
S ⟶ id = E	{ v = getId (id.name) write(v ‘=’ E.place) }
S ⟶ id[E] = E	{ v = getId (id.name) write(v + ‘#’ + E.place ‘=’ E1.place) }
E ⟶ id	{ v = getId (id.name) E.place = v }
E ⟶ id[E]	{ v = getId (id.name) E.place = v + ‘#’ + E1.place }
E ⟶ int	{ E.place = Constant of integer }
E ⟶ bitstring	{ E.place = Constant of bitstring }
E ⟶ boolean	{ E.place = Constant of boolean }
E ⟶ bitstrtoint	{ E.place = newtemp; write(E.place ‘=’ ‘bitstrtoint’ ‘(‘ E1.place ‘)’ }
E ⟶ sbitstrtoint	{ E.place = newtemp; write(E.place ‘=’ ‘sbitstrtoint’ ‘(‘ E1.place ‘)’ }
E ⟶ inttobitstr	{ E.place = newtemp; write(E.place ‘=’ ‘inttobitstr’ ‘(‘ E1.place ‘)’ }

ตารางที่ 4.4 (ก) การใช้กฎการแปลความหมายสร้างรูปแบบคำสั่งทรีแอดเดรสโค้ด

โปรดักชัน	กฎการแปลความหมาย
E ⟶ inttosbitstr	{ E.place = newtemp; write(E.place '=' 'inttosbitstr' '(' E1.place ')' }
E ⟶ bitadd	{ E.place = newtemp; write(E.place '=' 'bitadd' '(' E1.place ')' }
E ⟶ digitget	{ E.place = newtemp; write(E.place '=' 'digitget' E1.place }

E ⟶ call id	{ E.place = newtemp;
	for each item p on stack
	write('param' p);
	write('call id.place');
	next
	write(E.place '=' '#return')
	}

E ⟶ E1 + E2	{ E.place = newtemp; write(E.place '=' E1.place '+' E2.place) }
E ⟶ E1 - E2	{ E.place = newtemp; write(E.place '=' E1.place '-' E2.place) }
E ⟶ E1 * E2	{ E.place = newtemp; write(E.place '=' E1.place '*' E2.place) }
E ⟶ E1 / E2	{ E.place = newtemp; write(E.place '=' E1.place '/' E2.place) }
E ⟶ E1 and E2	{ E.place = newtemp; write(E.place '=' E1.place 'and' E2.place) }

ตารางที่ 4.4 (ข) การใช้กฎการแปลความหมายสร้างรูปแบบคำสั่งทรีแอดเดรสโค้ด

โปรดักชัน	กฎการแปลความหมาย
E ⟶ E1 or E2	{ E.place = newtemp; write(E.place '=' E1.place 'or' E2.place) }
E ⟶ E1 xor E2	{ E.place = newtemp; write(E.place '=' E1.place 'xor' E2.place) }
E ⟶ not E1	{ E.place = newtemp; write(E.place '=' 'not' E1.place) }
E ⟶ E1 > E2	{ E.place = newtemp; write(E.place '=' E1.place '>' E2.place) }
E ⟶ E1 < E2	{ E.place = newtemp; write(E.place '=' E1.place '<' E2.place) }
E ⟶ E1 >= E2	{ E.place = newtemp; write(E.place '=' E1.place '>=' E2.place) }
E ⟶ E1 <= E2	{ E.place = newtemp; write(E.place '=' E1.place '<=' E2.place) }
E ⟶ E1 == E2	{ E.place = newtemp; write(E.place '=' E1.place '==' E2.place) }
E ⟶ E1 != E2	{ E.place = newtemp; write(E.place '=' E1.place '!=' E2.place) }

ตารางที่ 4.4 (ค) การใช้กฎการแปลความหมายสร้างรูปแบบคำสั่งทรีแอดเดรสโค้ด

โปรดักชัน	กฎการแปลความหมาย
E ⟶ (E1)	{ E.place = newtemp; write(E.place = E1.place) }
E ⟶ E1 & E2	{ E.place = newtemp; write(E.place '=' E1.place '&' E2.place) }
E ⟶ E1 | E2	{ E.place = newtemp; write(E.place '=' E1.place '|' E2.place) }
E ⟶ E1 ^ E2	{ E.place = newtemp; write(E.place '=' E1.place '^' E2.place) }
E ⟶ ~ E1	{ E.place = newtemp; write(E.place '=' '~' E1.place) }
E ⟶ E1 << E2	{ E.place = newtemp; write(E.place '=' E1.place '<<' E2.place) }
E ⟶ E1 >> E2	{ E.place = newtemp; write(E.place '=' E1.place '>>' E2.place) }
E ⟶ E1 <<< E2	{ E.place = newtemp; write(E.place '=' E1.place '<<<' E2.place) }

ตารางที่ 4.4 (ง) การใช้กฎการแปลความหมายสร้างรูปแบบคำสั่งทรีแอดเดรสโค้ด

โปรดักชัน	กฎการแปลความหมาย
S ⟶ if E { S1 }	{ E.true = newlabel ; E.false = newlabel; E.place = newtemp; write(E.place '=' E.code) || write('if' E.place '=' 'true' gen(GOTO E.true ':') || gen(GOTO E.false ':') || gen(E.true ':') || s1.code || gen(E.false ':') }
S ⟶ if E { S1 } else { S2 }	{ E.true = newlabel ; E.false = newlabel; E.place = newtemp; write(E.place '=' E.code) || write('if' E.place '=' 'true' 'GOTO' E.true ':') || s2.code || write(GOTO E.false ':') || write(E.true ':') || s1.code || write(E.false ':') }
S ⟶ while E { S1 }	{ S.begin = newlabel; E.true = newlabel ; E.false = newlabel; E.place = newtemp; write(E.true ':') || write(E.place '=' E.code) || write('if' E.place '=' 'true' 'GOTO' E.true ':') || write('GOTO' E.false ':') || write(E.true ':') || s1.code || write('GOTO' S.begin ':') || write(E.false ':') }

D ⟶ T id	{ id.type = T.type
	write('dim' id.name);
	}

ตารางที่ 4.4 (จ) การใช้กฎการแปลความหมายสร้างรูปแบบคำสั่งทรีแอดเดรสโค้ด

โปรดักชัน	กฎการแปลความหมาย

D ⟶ T id[E]	{ id.type = T.type
	for index = 1 to E.place
	write('dim' id.name '#' index);
	next
	}

T ⟶ int	T.type = integer;
T ⟶ boolean	T.type = boolean;
T ⟶ string	T.type = bitstring;

การแปลงคำสั่งอินเตอร์มิเดียท จะใช้ตารางที่ 4.4 สร้างรูปแบบคำสั่งขณะท่องในโหนดไวยากรณ์ต้นไม้ OASTranslator จะสร้างอ็อบเจกต์ Translation โดยใช้ symbolTable เป็นพารามิเตอร์ส่งผ่านการเรียกใช้ด้วยเมทธอด apply () ซึ่งจะเรียกใช้โอเวอร์โรดเมทธอดที่ถูกเขียนอยู่ภายในคลาส Translation จากกฎการแปลความหมายจะใช้สำหรับการการแปลงคำสั่งกำหนดค่า, การประกาศตัวแปร, การประกาศฟังก์ชัน, คำสั่งทางเลือกและคำสั่งควบคุมลำดับการประมวลผล โดยนอนเทอร์มินอล E มีคุณสมบัติคือ

E.place ใช้เก็บค่าตัวสัญลักษณ์ E

E.code ใช้เก็บประโยคคำสั่งทรีแอดเดรสโค้ด สำหรับสัญลักษณ์ E ตัวนั้น

จากตารางที่ 4.4 id = E หรือ id [E] = E คือ ตัวแปรทั่วไป หรือตัวแปรแถวลำดับที่ถูกแทนค่าด้วย E โดยที่ E อาจจะเป็นค่าที่มาจาก นิพจน์การคำนวน, ฟังก์ชัน, ค่าคงที่ หรือ id ตัวอื่นๆ ประกอบด้วยโค้ดที่ใช้ในการคำนวนค่า E และเก็บค่าไว้ในตัวแปรชั่วคราว $t ใดๆ

newtemp คือค่าที่ถูกสร้างจากคลาส Temp เพื่อสร้างตัวเก็บข้อมูลชั่วคราวเป็นลำดับของชื่อเฉพาะ เช่น $t1, $t2, $t3, $t4 เป็นต้น เรียงตามลำดับตัวเลขโดยมีค่าไม่ซ้ำกัน

การแปลงประโยคสำหรับการประกาศตัวแปร D ⟶ T id หรือ D ⟶ T id [E] นอนเทอร์มินอล T มีคุณสมบัติ T.type ใช้เก็บชนิดของตัวแปร ตามด้วยตัวแปรทั่วไป หรือตัวแปรแบบแถวลำดับ E คือ ค่าคงที่ โดยที่ id.name คือชื่อตัวแปร กรณีตัวแปรชนิดแถวลำดับจะตามด้วยเครื่องหมาย '#' และลำดับของดัชนี

การแปลงคำสั่งควบคุมลำดับการประมวลผลในแต่ละโปรดักชัน โดยที่ E คือนิพจน์บูลลีนที่จะต้องถูกแปลง นิพจน์บูลลีน E แต่ละตัวจะมีอยู่สองเลเบลที่เกี่ยวข้องด้วยคือ E.true เป็นเลเบลที่ควบคุมลำดับการประมวลผลเมื่อค่าของนิพจน์ E เป็นจริง และ E.false เป็นเลเบลที่ควบคุมลำดับการประมวลผลเมื่อค่าของนิพจน์ E เป็นเท็จ โดยเลเบลเป็นสัญลักษณ์ตัวใหม่ทุกครั้งที่เรียกใช้สร้างจาก คลาส Genlabel แทนด้วย 'newlabel'

การแปลงประโยคสำหรับโปรดักชัน S ⟶ if E { S1 } จะสร้างเลเบลตัวใหม่เก็บไว้ E.true และ E.false จากนั้นสร้างตัวแปรชั่วคราวเก็บไว้ใน E.place และกำหนดค่าจาก E.code ให้ตัวแปรชั่วคราว เพื่อสร้างจุดกระโดดของคำสั่งไปยัง E.true ถ้า E มีค่าเป็นจริงตามด้วยประโยคคำสั่ง s1.code

การแปลงประโยคสำหรับโปรดักชัน S ⟶ if E { S1 } else { S2 } จะสร้างเลเบลตัวใหม่เก็บไว้ E.true และ E.false จากนั้นสร้างตัวแปรชั่วคราวเก็บไว้ใน E.place และกำหนดค่าจาก E.code ให้ตัวแปรชั่วคราว เพื่อสร้างจุดกระโดดของคำสั่งไปยัง E.true ถ้า E มีค่าเป็นจริงตามด้วยประโยคคำสั่ง s1.code ประโยคคำสั่ง s2.code ตามด้วย GOTO E.false ':' เป็นการสร้างจุดกระโดดคำสั่งเมื่อจบทำงานในส่วนของ s2.code

การแปลงประโยคสำหรับโปรดักชัน S ⟶ while E { S1 } จะสร้างเลเบลตัวใหม่เก็บไว้ใน S.begin เพื่อสร้างจุดวนกลับ จากนั้นสร้างเลเบลค่าของนิพจน์ E ที่เป็นจริงด้วย E.true และเลเบลค่าของนิพจน์ E ที่เป็นเท็จด้วย E.false สร้างตัวแปรชั่วคราวเก็บไว้ใน E.place จากนั้นกำหนดค่าจาก E.code ให้ตัวแปรชั่วคราว และสร้างคำสั่ง 'if' เมื่อเงื่อนไขเป็นจริงสร้างจุดกระโดดด้วย 'GOTO E.ture' ถ้าเงื่อนไขเป็นเท็จ สร้างจุดกระโดดด้วย 'GOTO E.false' จบคำสั่งใน S1.code สร้างคำสั่งจุดวนกลับด้วย 'GOTO S.begin'

4.2 ส่วนการสร้างภาพมโนทัศน์

เป็นส่วนสำหรับประมวลผล และแสดงภาพมโนทัศน์การทำงานของอัลกอริทึมแบ่งการออกแบบเป็น 2 ส่วน ดังนี้

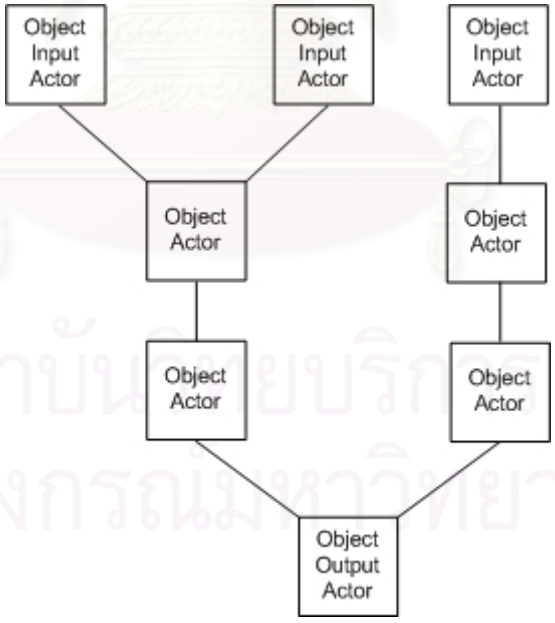
4.2.1 อินเตอร์มิเดียทอินเทอร์พรีเตอร์

4.2.2 ส่วนตัวประมวลผลสร้างภาพมโนทัศน์

4.2.1 อินเตอร์มิเดียทอินเทอร์พรีเตอร์

การทำงานส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ มีหน้าที่สร้างโครงสร้างข้อมูล (Animation Data Structure) และคำสั่ง (Animation Instruction) สำหรับแสดงภาพมโนทัศน์ การทำงานประกอบด้วย 2 ส่วน ดังนี้

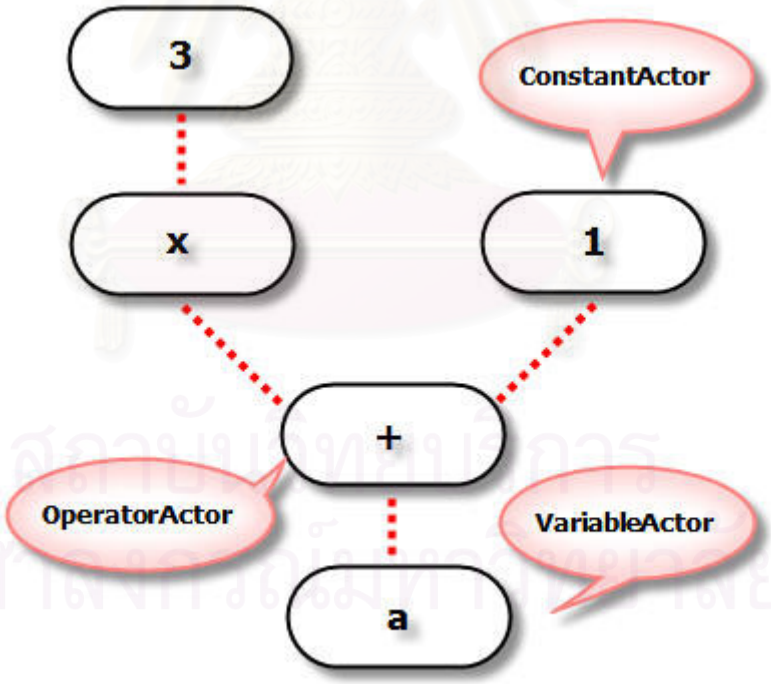
1. การแปลงคำสั่งอินเตอร์มิเดียทเป็นโครงสร้างข้อมูลสำหรับแสดงภาพมโนทัศน์ ส่งให้ส่วนประมวลผลสร้างภาพมโนทัศน์ นำไปใช้วาดภาพโครงสร้างข้อมูลความสัมพันธ์ โดยมองในลักษณะความสัมพันธ์จากล่างขึ้นบน ดังรูปที่ 4.15



รูปที่ 4.15 โครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์มุมมองจากล่างขึ้นบน

โครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์ คือ โครงสร้างข้อมูลที่มีความสัมพันธ์กันโดยพิจาราณาจากตัวแปรผลลัพธ์เป็นหลัก การกระทำใดที่เกี่ยวข้องกับการเปลี่ยนแปลงค่าของตัวแปรผลลัพธ์จะถูกนำมาสร้างความสัมพันธ์ในมุมมองโครงสร้างข้อมูลต้นไม้ เป็นองค์ประกอบหลัก โดยพิจารณาจากโหนดแต่ละโหนด ซึ่งก็คือองค์ประกอบของโครงสร้างข้อมูลมีความสัมพันธ์กันในมุมมองจากล่างขึ้นบน ซึ่งเปรียบได้กับนักแสดงหรือแอคเตอร์ (Actor) มีหลายประเภทในหน่วยย่อยของภาษา เช่น ตัวแปร, ตัวแปรแบบแถวลำดับ, ตัวแปรนำเข้า, ตัวแปรผลลัพธ์, เส้นความสัมพันธ์ เป็นต้น

โหนดแต่ละโหนดมีไว้เพื่อเก็บแอคเตอร์ โดยที่แอคเตอร์แต่ละตัวจะถูกสร้างขึ้นในขั้นตอนการแปลคำสั่งอินเตอร์มิเดียท โดยส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ทำการแบ่งคำที่สนใจภายในแต่ละคำสั่งนั้นคือ องค์ประกอบที่มีความสัมพันธ์กับตัวแปรผลลัพธ์เพื่อ นำมาสร้างความสัมพันธ์ โดยเริ่มต้นจากแอคเตอร์ Output หรือตัวแปรผลลัพธ์ขึ้นไปจนถึงแอคเตอร์ Input หรือตัวแปรนำเข้า



รูปที่ 4.16 ตัวอย่างโครงสร้างข้อมูลสำหรับใช้แสดงภาพมโนทัศน์

โดยที่ความสัมพันธ์เหล่านี้จะถูกนำมาใช้เมื่อผู้ใช้สั่งให้มีการแสดงภาพมโนทัศน์ตัวอย่าง ความสัมพันธ์ระหว่างแอคเตอร์ เช่น

```
x = 3;
a = x + 1;
```

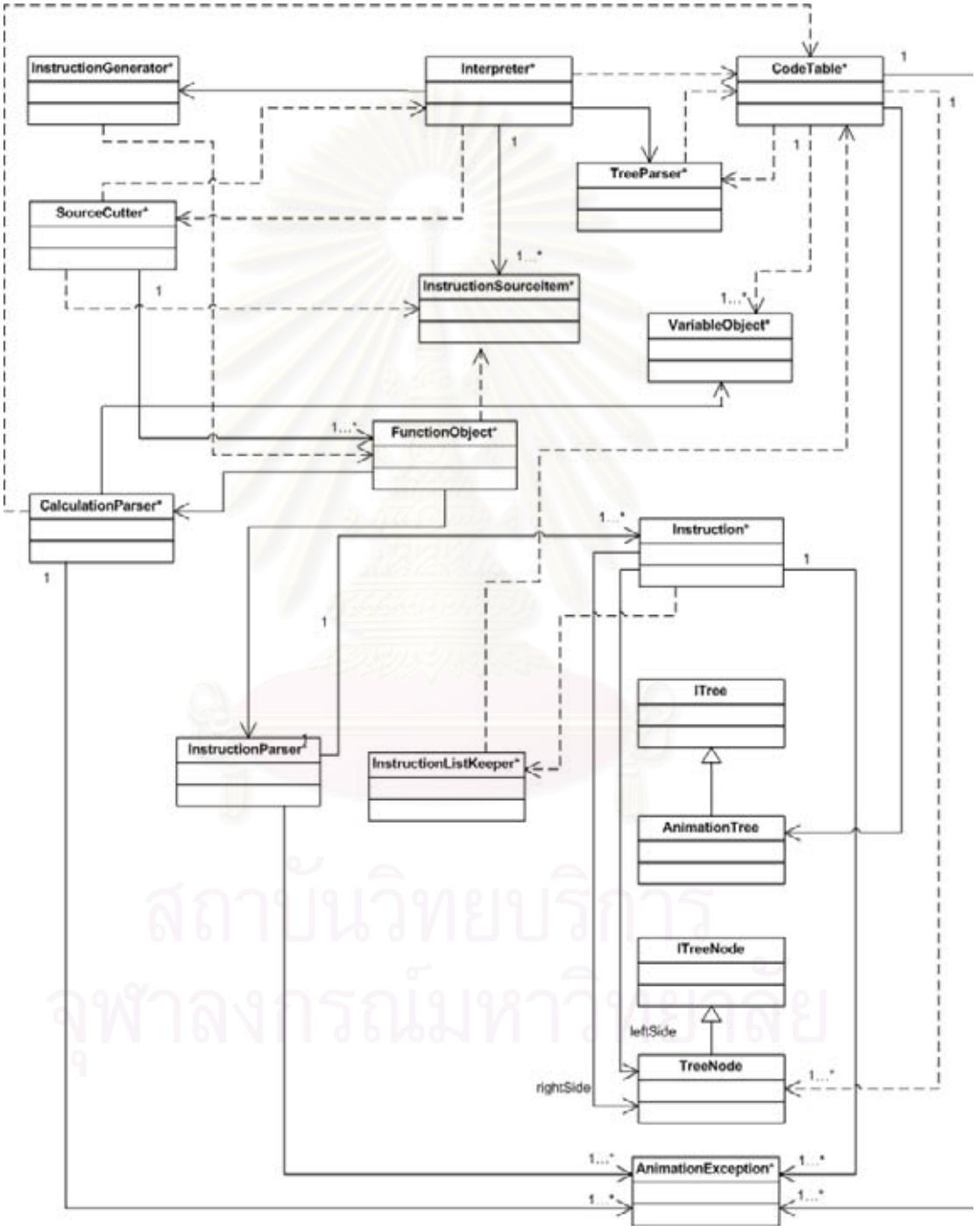
จากรูปที่ 4.16 โหนดราก ในกรณีนี้คือ ตัวแปร a ซึ่งจะถูกสร้างเป็นแอคเตอร์ชนิดตัวแปร (VariableActor) โดยที่ a มีโหนดลูกคือโอเปอร์เรเตอร์บวก ซึ่งเป็นแอคเตอร์อีกชนิดหนึ่งที่ชื่อว่าแอคเตอร์ชนิดโอเปอร์เรเตอร์ (OperatorActor) ซึ่งมีลูกเป็นตัวแปร x และค่าคงที่ 1 ตามลำดับ

2. การแปลงคำสั่งอินเตอร์มิเดียทเป็นคำสั่งสำหรับสร้างภาพมโนทัศน์ ขั้นตอนนี้จะทำหน้าที่ส่งโครงสร้างข้อมูล และคำสั่งสำหรับสร้างภาพมโนทัศน์ให้ส่วนประมวลผลสร้างภาพมโนทัศน์ ในส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ ผู้วิจัยเลือกใช้ Grammartica เป็นเครื่องมือช่วยแปลความหมายคำสั่งอินเตอร์มิเดียททีละคำสั่ง เนื่องจากเป็นเครื่องมือที่พัฒนาด้วยภาษาจาวา เช่นเดียวกับเครื่องมือสร้างตัวแปลภาษา การออกแบบไวยากรณ์มีความชัดเจนและง่ายในการนำมาใช้งาน และยังสามารถทดสอบการทำงาน และทดสอบความถูกต้องของไวยากรณ์ได้ในสถานะรันไทม์ (Runtime) ก่อนนำไปพัฒนาใช้งานจริง การใช้งานของเครื่องมือนำมาเพื่อแยกคำที่สนใจ เพื่อสร้างคลาสตรวจสอบไวยากรณ์คำสั่งอินเตอร์มิเดียท รายละเอียดการบรรยายคำ และบรรยายไวยากรณ์ แสดงในภาคผนวก ข. โดยเขียนแอคชั่นโค้ดเพื่อใช้ทำหน้าที่ดังนี้

- สร้างอ็อบเจกต์แอคเตอร์และกำหนดความสร้างสัมพันธ์ของแอคเตอร์ภายในโครงสร้างข้อมูลสำหรับการสร้างภาพมโนทัศน์
- คำนวณค่าตัวแปรในคำสั่งอินเตอร์มิเดียท
- สร้างคำสั่งสำหรับแสดงภาพมโนทัศน์

แผนภาพคลาสส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์

แผนภาพคลาสใช้แสดงคลาสและความสัมพันธ์ระหว่างคลาสต่างๆ เพื่อจำลองภาพการออกแบบส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ ความสัมพันธ์ระหว่างวัตถุ (Object) ดังรูปที่ 4.17



รูปที่ 4.17 ความสัมพันธ์ของคลาสในส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์

คลาสที่เกี่ยวข้องกับอินเตอร์มิเดียทอินเทอร์พรีเตอร์ประกอบด้วยคลาสที่สำคัญดังต่อไปนี้

- คลาส Interpreter
- คลาส SourceCutter
- คลาส InstructionSourceItem
- คลาส CodeTable
- คลาส ITreeNode
- คลาส TreeNode
- คลาส TreeParser
- คลาส ITree
- คลาส AnimationTree
- คลาส InstructionGenerator
- คลาส VariableObject
- คลาส FunctionObject
- คลาส CalculationParser
- คลาส InstructionParser
- คลาส Instruction
- คลาส InstructionListKeeper
- คลาส AnimationException

คลาส Interpreter คือ คลาสควบคุมการทำงานในส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ เพื่อรับแฟ้มข้อมูลคำสั่งอินเตอร์มิเดียทจากส่วนการแปลภาษาเพื่อนำไปใช้สร้างโครงสร้างข้อมูลและคำสั่งสำหรับสร้างภาพมโนทัศน์

คลาส SourceCutter คือ คลาสที่ทำหน้าที่แปลงแฟ้มข้อมูลคำสั่งอินเตอร์มิเดียทแต่ละบรรทัดเป็นอ็อบเจกต์ InstructionSourceItem โดยที่หนึ่งบรรทัดของคำสั่งอินเตอร์มิเดียทจะถูก

แปลงเป็นหนึ่งอ็อบเจกต์ InstructionSourceItem และทำหน้าที่สร้างอ็อบเจกต์ FunctionObject ในขั้นตอนการประมวลผลคำสั่งอินเตอร์มิเดียท

คลาส InstructionSourceItem คือ คลาสที่เก็บข้อมูลรายละเอียดของคำสั่งอินเตอร์มิเดียท ซึ่งแต่ละบรรทัดของคำสั่งอินเตอร์มิเดียทจะมีข้อมูลที่จำเป็นต่อการสร้างความสัมพันธ์ของตัวแปร โดยจะแบ่งแยกการจัดเก็บข้อมูลแต่ละบรรทัด ดังนี้

- คำสั่งอินเตอร์มิเดียท
- หมายเลขบรรทัดของโปรแกรมต้นฉบับ
- ระดับคำสั่งอินเตอร์มิเดียท
- หมายเลขบรรทัดของคำสั่งอินเตอร์มิเดียท

คลาส CodeTable คือ คลาสทำหน้าที่เก็บอ็อบเจกต์แอคเตอร์ทุกตัวที่ใช้วาดลงบนส่วนแสดงภาพมโนทัศน์ซึ่งถูกสร้างจากคลาส TreeParser และใช้เก็บอ็อบเจกต์ตัวแปร VariableObject ซึ่งถูกสร้างจากคลาส CalculationParser ในขั้นตอนการประมวลผลคำสั่งอินเตอร์มิเดียท

คลาส ITreeNode คือ อินเตอร์เฟซคลาสที่กำหนดนิยามของ TreeNode โดยจะกำหนดเมทธอดที่นิยามความหมายของ TreeNode เช่น

- getParent เพื่อหาว่า ITreeNode ตัวนี้มี ITreeNode ตัวใดเป็นโหนดแม่
- getChildren เพื่อหาว่า ITreeNode ตัวนี้มี ITreeNode ตัวใดบ้างที่เป็นโหนดลูก

คลาส TreeNode คือ คลาสทำหน้าที่เก็บข้อมูลแอคเตอร์เพื่อให้ส่วนประมวลผลการสร้างภาพมโนทัศน์ นำความสัมพันธ์ของโครงสร้างข้อมูลนี้ไปวาดลงบนส่วนแสดงภาพมโนทัศน์ โดยในแต่ละ TreeNode จะประกอบไปด้วย ฟิลด์ ที่มีรายละเอียดดังนี้

- ฟิลด์ Actor เพื่อบอกว่าจะโชว์รูปและโชว์ชื่ออะไรในการแสดง
- ฟิลด์ Parent เพื่อบอกว่า TreeNode ตัวนี้มี TreeNode ตัวใดเป็นโหนดแม่
- ฟิลด์ Children เพื่อบอกว่า TreeNode ตัวนี้มี TreeNode ตัวใดเป็นโหนดลูก

คลาส TreeParser คือคลาสที่สืบทอดมาจากคลาส IcodeGrammarAnalyzer สร้างโดย Grammatica ทำหน้าที่สร้าง TreeNode และกำหนดความสัมพันธ์ของแต่ละโหนดว่า TreeNode ตัวใดเป็นโหนดแม่และ TreeNode ตัวใดเป็นโหนดลูก

คลาส ITree คือ อินเตอร์เฟซคลาสที่กำหนดนิยามของ Tree โดยจะกำหนดเมทธอดที่สำคัญนิยามความหมายของ Tree มีรายละเอียดดังนี้

- getRoot เพื่อหาว่าใน ITree ตัวนี้มี ITreeNode ตัวใดเป็นโหนดราก

- setRoot เพื่อกำหนดโหนดรากให้กับ ITree

คลาส AnimationTree คือ คลาสทำหน้าที่เก็บตำแหน่ง และความสัมพันธ์ของแอคเตอร์ที่อยู่ใน TreeNode โดยที่คลาส AnimationTree จะถูกเรียกใช้จากคลาส AnimationStage นำข้อมูลของแอคเตอร์ที่เก็บอยู่นำไปใช้ในการสร้างภาพบนส่วนแสดงภาพมโนทัศน์ โดยใช้คลังโปรแกรมของ Java Swing เมทธอดที่สำคัญมีดังนี้

- traverse (ITreeNode currentNode) ทำหน้าที่ท่องโหนดใน AnimationTree แบบเรียกซ้ำ (recursive) เพื่อที่จะนำข้อมูลของแอคเตอร์เก็บไว้เพื่อเรียกใช้งานต่อไป

- allocateConnectors () ทำหน้าที่หาความสัมพันธ์ของแอคเตอร์แต่ละตัว เพื่อสร้างแอคเตอร์ Connector

- assignPositionToActorsBottomUp () ทำหน้าที่กำหนดตำแหน่งของแอคเตอร์แต่ละตัวที่วาดลงบนพื้นที่ส่วนแสดงภาพมโนทัศน์

- getAllInstructions () ทำหน้าที่กำหนดคำสั่งแสดงภาพมโนทัศน์ให้กับแอคเตอร์ที่อยู่ภายในโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์

คลาส InstructionGenerator เป็นคลาสสำหรับสั่งสร้าง FunctionObject โดยจะรับข้อมูลภาษาอินเตอร์มิเดียทส่งให้คลาส SourceCutter และทำหน้าที่เรียก FunctionObject global ซึ่งเป็นฟังก์ชันหลักใช้เริ่มต้นการทำงาน

คลาส Variable object คือ คลาสที่เก็บข้อมูลตัวแปร เพื่อนำไปใช้ในการคำนวณ ข้อมูลที่จัดเก็บประกอบด้วย ชื่อ, ประเภทของตัวแปร และค่าของตัวแปร

คลาส FunctionObject คือ คลาสที่เก็บรายการคำสั่งที่อยู่ภายในฟังก์ชัน และทำหน้าที่ส่งคำสั่งอินเตอร์มิเดียทให้กับคลาส CalculationParser และคลาส InstructionParser

คลาส CalculationParser คือ คลาสที่สืบทอดมาจาก IcodeGrammarAnalyzer สร้างโดย Grammatica เพื่อประมวลผลการทำงานในแต่ละคำสั่งอินเตอร์มิเดียจที่ได้รับจาก

FunctionObject โดยจะทำการปรับปรุงค่าการคำนวณของแต่ละตัวแปรที่ถูกจัดเก็บไว้ในอ็อบเจกต์ CodeTable

คลาส InstructionParser คือ คลาสที่สืบทอดมาจาก IcodeGrammarAnalyzer ที่สร้างโดย Grammatica ทำหน้าที่สร้างคำสั่งสำหรับการแสดงภาพมโนทัศน์

คลาส Instruction คือคลาสที่ใช้เก็บคำสั่งสำหรับแสดงภาพมโนทัศน์ ซึ่งเป็นคำสั่งที่มีไว้เพื่อกำหนดว่าแอคเตอร์ตัวใดบ้างที่จะต้องทำการแสดง โครงสร้างของคำสั่งประกอบด้วยรูปแบบ ดังนี้

- ฟิลด์ leftSide คือ TreeNode ด้านซ้ายของโอเปอเรเตอร์ ยกตัวอย่างเช่น x = 3 โดยที่ด้านซ้ายในที่นี้คือ x
- ฟิลด์ rightSide คือ TreeNode ด้านขวาของโอเปอเรเตอร์ ยกตัวอย่างเช่น x = 3 โดยที่ด้านขวาในที่นี้คือ 3
- ฟิลด์ transferValue คือ ค่าที่ใช้เป็นตัวเคลื่อนไหวจากด้านขวาไปด้านซ้าย ยกตัวอย่างเช่น x = 3 ค่าที่เป็นตัววิ่งในที่นี้คือ 3

ประเภทของ Instruction แบ่งออกได้เป็น 3 ประเภทดังต่อไปนี้

- คำสั่งทั่วไป คือคำสั่งกำหนดค่าที่มีตัวกำหนดค่าเพียงตัวเดียว
  ยกตัวอย่างเช่น x = 3; เมื่อแปลงเป็น Instruction จะมีความหมายดังนี้

  1) แอคเตอร์ที่มีประเภทเป็น ConstantActor และมีค่าข้อมูลคือ 3 ทำการแสดง (สัญลักษณ์เปลี่ยนจากสีแดงเป็นสีเขียว)

  2) แอคเตอร์ที่มีประเภทเป็น Connector ที่เชื่อมระหว่าง 3 กับ x ทำการแสดง (ค่า 3 เคลื่อนที่ไปที่ x)

  แอคเตอร์ที่มีประเภทเป็น VariableActor และค่าเป็น x ทำการแสดง (สัญลักษณ์เปลี่ยนจากสีแดงเป็นสีเขียว)

- คำสั่งซ้อน คือคำสั่งกำหนดค่า โดยที่ด้านขวาจะเป็นนิพจน์ใดๆ มีเพียงตัวดำเนินการเดียว ยกตัวอย่างเช่น x = a + b; ในกรณีนี้จะเกิดการเคลื่อนไหวของค่าสามครั้งด้วยกัน ได้แก่

  - ค่าที่เป็นตัววิ่งจากแอคเตอร์ a ไปหาแอคเตอร์โอเปอเรเตอร์บวก

- ค่าที่เป็นตัววิ่งจากแอคเตอร์ b ไปหาแอคเตอร์โอเปอเรเตอร์บวก
- ค่าที่เป็นตัววิ่งจากแอคเตอร์โอเปอเรเตอร์บวก ไปหาแอคเตอร์ x

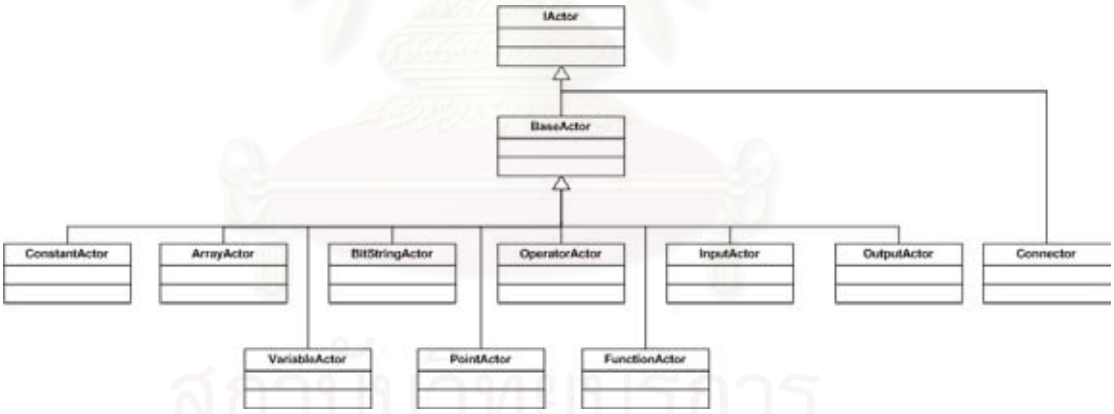
จะเห็นได้ว่าค่าที่เป็นตัววิ่งไม่จำเป็นที่จะต้องเท่ากันจึงต้องมีการกำหนดค่า transferValue ให้กับแอคเตอร์ a, b และ โอเปอเรเตอร์บวก

- คำสั่งเงื่อนไข คือคำสั่งที่ไม่ต้องแสดงค่าเคลื่อนไหว ดังนั้นจึงไม่ต้องมีการกำหนดค่า transferValue

คลาส InstructionListKeeper เป็นคลาสที่เก็บรายการคำสั่งสำหรับใช้แสดงภาพมโนทัศน์ทั้งหมดที่ได้รับจากคลาส InstructionParser

คลาส AnimationException เป็นคลาสสำหรับเก็บข้อผิดพลาดระหว่างการสร้างข้อมูลสำหรับแสดงภาพมโนทัศน์

คลาสแอคเตอร์หรือนักแสดงสืบทอดมาจากคลาส BaseActor ซึ่งเป็นคลาสแม่ของแอคเตอร์ทุกประเภทยกเว้นแอคเตอร์ตัวเชื่อม ดังรูปที่ 4.18



รูปที่ 4.18 ความสัมพันธ์ของคลาสแอคเตอร์กับคลาส BaseActor

รายละเอียดของคลาสที่เกี่ยวข้องกับแอคเตอร์ ประกอบด้วยคลาสต่างๆ ดังนี้

- คลาส IActor
- คลาส BaseActor
- คลาส InputActor

- คลาส OutputActor
- คลาส VariableActor
- คลาส OperatorActor
- คลาส ConstantActor
- คลาส ArrayActor
- คลาส BitStringActor
- คลาส FunctionActor
- คลาส PointActor

คลาส IActor คืออินเตอร์เฟซคลาสที่กำหนดความเป็นแอคเตอร์ ในคลาสแอคเตอร์ต่างๆ ทุกประเภทให้มีกระทำที่เหมือนกัน คือ

1) วาดตนเองลงบนผืนผ้ากราฟิก (Paint Actor) เนื่องจากแอคเตอร์ทุกตัวต้องรู้ว่า เมื่อส่วนแสดงภาพมโนทัศน์สั่งให้วาดแอคเตอร์จะต้องทำการวาดอย่างไร ซึ่งคำสั่งนี้จะถูกเรียกโดยอัตโนมัติตลอดเวลาเพราะเป็นข้อกำหนดของทุกคลาสที่สืบทอดมาจาก JComponent

2) ทำการแสดง (Action) เมื่อส่วนประมวลผลการแสดงภาพมโนทัศน์ ทำการสั่งให้แอคเตอร์ตัวปัจจุบันแสดงการทำงาน เมทธอดนี้จะทำหน้าที่กระทำการแสดงการทำงานของแอคเตอร์แต่ละประเภท ยกตัวอย่าง เช่น

- ถ้าหากแอคเตอร์เป็นประเภท InputActor เมื่อมีคำสั่งให้แสดงจะมีการกระพริบของดิจิตด้านซ้ายสุด จากนั้นดิจิตทางขวาที่เหลือจะเคลื่อนที่มาทางซ้ายหนึ่งตำแหน่ง

- ถ้าหากแอคเตอร์เป็นประเภท VariableActor เมื่อมีคำสั่งให้แสดงจะมีสีเขียวปรากฏขึ้นที่มุมซ้ายบนเพื่อแสดงให้ผู้ใช้เห็นว่ามีค่าเคลื่อนที่มายังแอคเตอร์ และจะเปลี่ยนกลับเป็นสีแดงหากค่าเคลื่อนที่ออกจากแอคเตอร์

3) บอกตำแหน่งปัจจุบันในแกน x, y ใช้สำหรับการตั้งต้นตำแหน่งให้แอคเตอร์แต่ละตัว เพื่อนำมาวาดบนพื้นที่ส่วนแสดงภาพมโนทัศน์

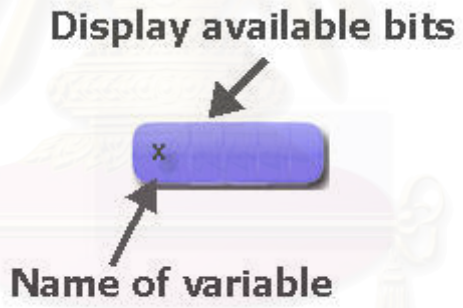
คลาส BaseActor คือคลาสที่ทำการสืบทอดจากอินเตอร์เฟซ IActor เนื่องจากแอคเตอร์เกือบทุกประเภทยกเว้นตัวเชื่อม (Connector) มีความคล้ายคลึงกันหลายประการ จึงใช้หลัก

polymorphism [17] เพื่อให้แอคเตอร์ทุกประเภทเป็นที่รู้จักในนามของ BaseActor ในกรณีที่มีการเรียกใช้จากโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์

BaseActor ยังมีหน้าที่แสดงผลในพื้นที่ส่วนแสดงภาพมโนทัศน์ และทำหน้าที่แสดงภาพมโนทัศน์เมื่อมีคำสั่งหรือ Instruction ระบุว่าแอคเตอร์ตัวนี้ต้องทำการแสดง เมทธอดที่สำคัญมีดังนี้

- คอนสตรัคเตอร์ (Constructor) เมื่อมีการประกาศวัตถุของคลาส BaseActor ตัวคลาสจะทำการสร้างตัวแปรประเภท JPanel และ Image เพื่อใช้ในการวาดของคลาสลูก
- paintActor (Graphics2D g2d) เป็นเมทธอดที่สืบทอดมาจากอินเตอร์เฟซ IActor โดยเมื่อมีคำสั่งให้วาดรูป แอคเตอร์จะทำการวาดตนเองลงบนผืนผ้ากราฟิก

คลาส InputActor สืบทอดมาจาก BaseActor โดยทำหน้าที่วาดเป็นตัวแปรนำเข้าในมุมมองของผู้ใช้ ตัวอย่างของ InputActor ดังรูปที่ 4.19

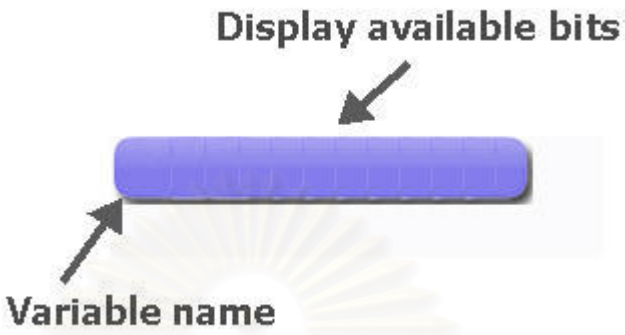


รูปที่ 4.19 InputActor

แอคเตอร์ประเภทนี้จะทำการวาดค่าของตัวแปรนำเข้า โดยที่ค่านี้ถูกกำหนดให้แสดงได้สี่หลัก หากตัวแปรนำเข้ามีค่ามากกว่า 4 หลัก จะเกิดเส้นประขึ้นด้านข้างเพื่อแสดงว่ามีค่าส่วนที่เหลือภายในตัวแปร

การแสดงของ InputActor เมื่อมีคำสั่งให้แสดงจะมีการหน่วงเวลาช่วงหนึ่ง (ขึ้นอยู่กับความเร็วที่ผู้ใช้กำหนดผ่านส่วนประสานงานผู้ใช้) หลังจากนั้นจะทำการเลื่อนค่าที่อยู่ด้านซ้ายสุดออก แล้วนำค่าถัดไปมาเป็นตำแหน่งแรกแทนที่

คลาส OutputActor สืบทอดมาจาก BaseActor โดยทำหน้าที่วาดรูปตัวแปรผลลัพธ์ในมุมมองของผู้ใช้ ตัวอย่างของ OutputActor ดังรูปที่ 4.20



รูปที่ 4.20 OutputActor

แอคเตอร์ประเภทนี้จะทำการวาดค่าปัจจุบันของตัวแปรผลลัพธ์ โดยที่ค่านี้ถูกกำหนดให้แสดงได้สิบหลัก หากตัวแปรผลลัพธ์มีค่ามากกว่าสิบหลักจะเกิดเส้นประขึ้นเพื่อแสดงว่าเพื่อแสดงว่ายังมีค่าส่วนที่เหลือภายในตัวแปร การแสดงของ OutputActor จะทำการเลื่อนไปทางขวาครั้งละหนึ่งหลักเมื่อมีการกำหนดค่าให้กับตัวแปรผลลัพธ์

คลาส VariableActor สืบทอดมาจาก BaseActor โดยทำหน้าที่วาดรูปตัวแปรทั่วไปในมุมมองของผู้ใช้ ตัวอย่างของ VariableActor ดังรูปที่ 4.21

รูปที่ 4.21 VariableActor

แอคเตอร์ประเภทนี้จะทำการวาดชื่อของตัวแปรทั่วไปตรงช่องว่าง โดยที่แอคเตอร์ประเภทนี้จะมีอยู่สองสถานะคือ แสดง (Active) หรือหยุดการแสดง (Inactive) สถานะแสดงอยู่หากมองในมุมของผู้ใช้ คือ เมื่อมีค่าเคลื่อนที่เข้ามาในแอคเตอร์ หลังจากนั้นจะเปลี่ยนเป็นสถานะหยุดการแสดง เมื่อค่าเคลื่อนที่ออกจากแอคเตอร์

คลาส OperatorActor สืบทอดมาจาก BaseActor โดยทำหน้าที่วาดเป็นตัวดำเนินการในมุมมองของผู้ใช้ ตัวดำเนินการมีทั้งหมด 21 ประเภท ดังรูปที่ 4.22

รูปที่ 4.22 OperatorActor

แอคเตอร์ประเภทนี้จะทำการวาดตัวดำเนินการ ซึ่งแบ่งเป็นสามประเภทใหญ่ๆ ได้แก่

- ตัวดำเนินการที่กระทำกับสายอักขระบิต
- ตัวดำเนินการทางตรรกกะ
- ตัวดำเนินการทางคณิตศาสตร์

OperatorActor ไม่มีการแสดงใดๆ ดังนั้นเมื่อส่วนประมวลผลภาพมโนทัศน์ออกคำสั่ง จะข้ามการแสดงของแอคเตอร์ประเภทนี้ไป

คลาส ConstantActor สืบทอดมาจาก BaseActor โดยทำหน้าที่วาดเป็นค่าคงที่ในมุมมองของผู้ใช้ ตัวอย่างของ ConstantActor ดังรูปที่ 4.23

รูปที่ 4.23 ConstantActor

คลาส ArrayActor สืบทอดมาจาก BaseActor โดยทำหน้าที่วาดเป็นตัวแปรอาเรย์ในมุมมองของผู้ใช้ ตัวอย่างของ ArrayActor ดังรูปที่ 4.24

รูปที่ 4.24 ArrayActor

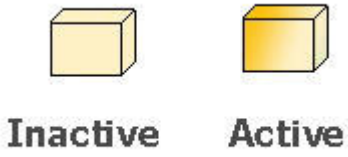
แอคเตอร์ประเภทนี้จะทำการวาดชื่อของตัวแปรตรงช่องว่างตามด้วยก้ามปู และตัวเลขบอกตำแหน่ง ยกตัวอย่าง เช่น temp [1]

คลาส BitStringActor สืบทอดมาจาก BaseActor โดยทำหน้าที่วาดเป็นตัวแปรบิตสตริงในมุมมองของผู้ใช้ ตัวอย่างของ BitStringActor ดังรูปที่ 4.25

รูปที่ 4.25 BitStringActor

เนื่องจากตัวแปรแบบสายอักขระบิตเป็นได้ทั้งตัวแปรเดี่ยว และสามารถอ้างอิงลำดับได้ ในกรณีที่เป็นตัวแปรเดี่ยวแอคเตอร์ประเภทนี้จะทำการวาดชื่อของตัวแปรในที่ว่างคล้ายกับ VariableActor ในกรณีที่เป็นอาเรย์แอคเตอร์ประเภทนี้จะทำการวาดชื่อของตัวแปรตรงช่องว่างตามด้วยก้ามปูและตัวเลขบอกตำแหน่ง ยกตัวอย่างเช่น bit [0] เป็นต้น

คลาส FunctionActor สืบทอดมาจาก BaseActor โดยทำหน้าที่วาดรูปฟังก์ชั่นสำหรับใช้แทนยูสเซอร์ฟังก์ชันหรือฟังก์ชันภายใน โดยไม่แสดงขั้นตอนการทำงานภายในฟังก์ชัน ตัวอย่างของ FunctionActor ดังรูปที่ 4.26



รูปที่ 4.26 FunctionActor

คลาส PointActor สืบทอดมาจาก BaseActor โดยทำหน้าที่วาดเป็นจุดเพื่อเชื่อมต่อ ตัวอย่างของ PointActor ดังรูปที่ 4.27

รูปที่ 4.27 PointActor

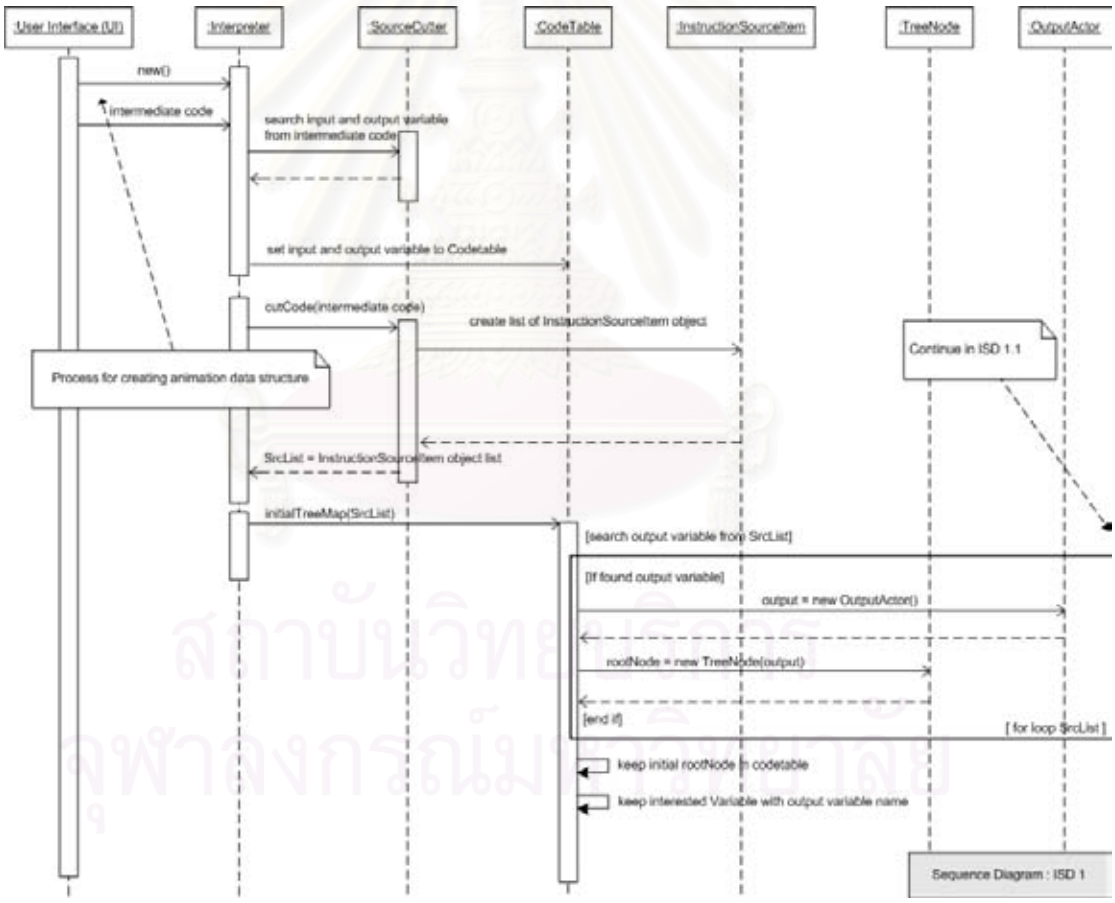
เนื่องจากจุดเป็นเพียงสิ่งสมมุติที่ช่วยในสร้างภาพมโนทัศน์ PointActor ไม่มีการแสดงใดๆ ดังนั้นเมื่อส่วนประมวลผลภาพมโนทัศน์ออกคำสั่ง จะข้ามการแสดงของแอคเตอร์ประเภทนี้ไป

แผนภาพซีเควนซ์แสดงการทำงานส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์มีลำดับของกิจกรรม ดังนี้

1. แปลงคำสั่งอินเตอร์มิเดียท เป็นโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์

การทำงานในขั้นตอนนี้จะรับคำสั่งอินเตอร์มิเดียท นำเข้ามาประมวลผลการแสดงภาพมโนทัศน์มีขั้นตอนการทำงานดังนี้

1.1 ส่วนประสานงานผู้ใช้ หรือ UI สร้าง Interpreter โดยใช้คำสั่งอินเตอร์มิเดียทเป็นพารามิเตอร์ การทำงานจะค้นหาตัวแปรนำเข้า และตัวแปรผลลัพธ์จากคำสั่งอินเตอร์มิเดียท เป็นค่าตั้งต้นจัดเก็บค่าไว้ใน CodeTable ดังรูปที่ 4.28

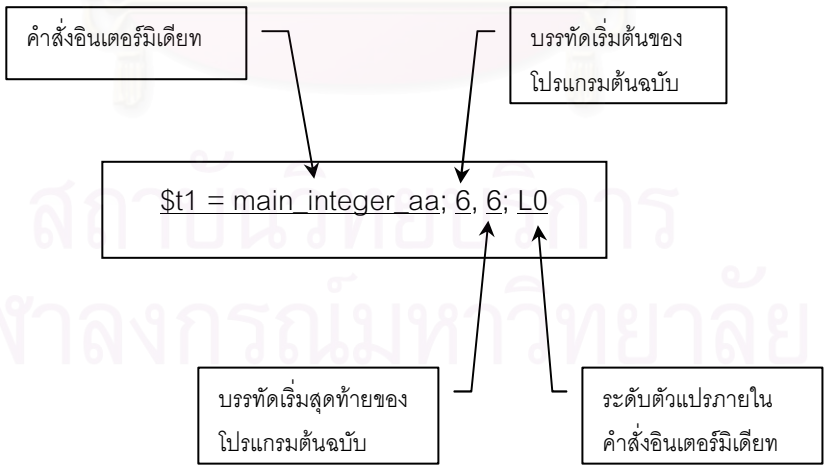


รูปที่ 4.28 แผนภาพซีเควนซ์แสดงการทำงานส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์

1.2 Interpreter เรียกใช้เมทธอด cutCode () เพื่อแปลงคำสั่งอินเตอร์มิเดียทแต่ละบรรทัดเป็น InstructionSourceItem เพื่อใช้แยกความแตกต่างของข้อมูลที่ได้รับและง่ายในการนำไปใช้งานในขั้นตอนอื่นๆ ต่อไป โดยทำหน้าที่แยกข้อมูลในแต่ละบบรทัดออกเป็นส่วนๆ ดังนี้

- คำสั่งอินเตอร์มิเดียท
- บรรทัดเริ่มต้นของโปรแกรมต้นฉบับที่ถูกนำมาแปลงเป็นคำสั่งอินเตอร์มิเดียท
- บรรทัดสุดท้ายของโปรแกรมต้นฉบับที่ถูกนำมาแปลงเป็นคำสั่งอินเตอร์มิเดียท
- ระดับตัวแปรภายในคำสั่งอินเตอร์มิเดียท โดยค่าของระดับจะเป็นตัวบอกว่าคำสั่งอินเตอร์มิเดียทบรรทัดนี้อยู่ในคำสั่งเงื่อนไข หรือคำสั่งทางเลือกใดในภาษาอินเตอร์มิเดียท โดยที่ตัวแปรใดที่อยู่ภายใต้คำสั่งเหล่านี้จะมีค่าระดับการอ้างอิงกับตัวแปรอื่นที่เกี่ยวข้องก่อนหน้านี้เพื่อใช้สร้างความสัมพันธ์ของตัวแปรจากภายนอก โดยที่ค่าระดับการอ้างอิงตัวแปรนี้จะเพิ่มค่าขึ้นถ้ามีคำสั่งเงื่อนไขหรือคำสั่งทางเลือกเพิ่มขึ้น ค่าระดับนี้ก็จะเพิ่มตามไปด้วย แต่ละส่วนจะถูกขั้นด้วยเครื่องหมาย “;” และ “,” ใช้แยกบรรทัดเริ่มต้นและสุดท้ายของโปรแกรมต้นฉบับออกจากกัน

ตัวอย่าง การแยกข้อมูลแต่ละบรรทัดในคำสั่งอินเตอร์มิเดียท

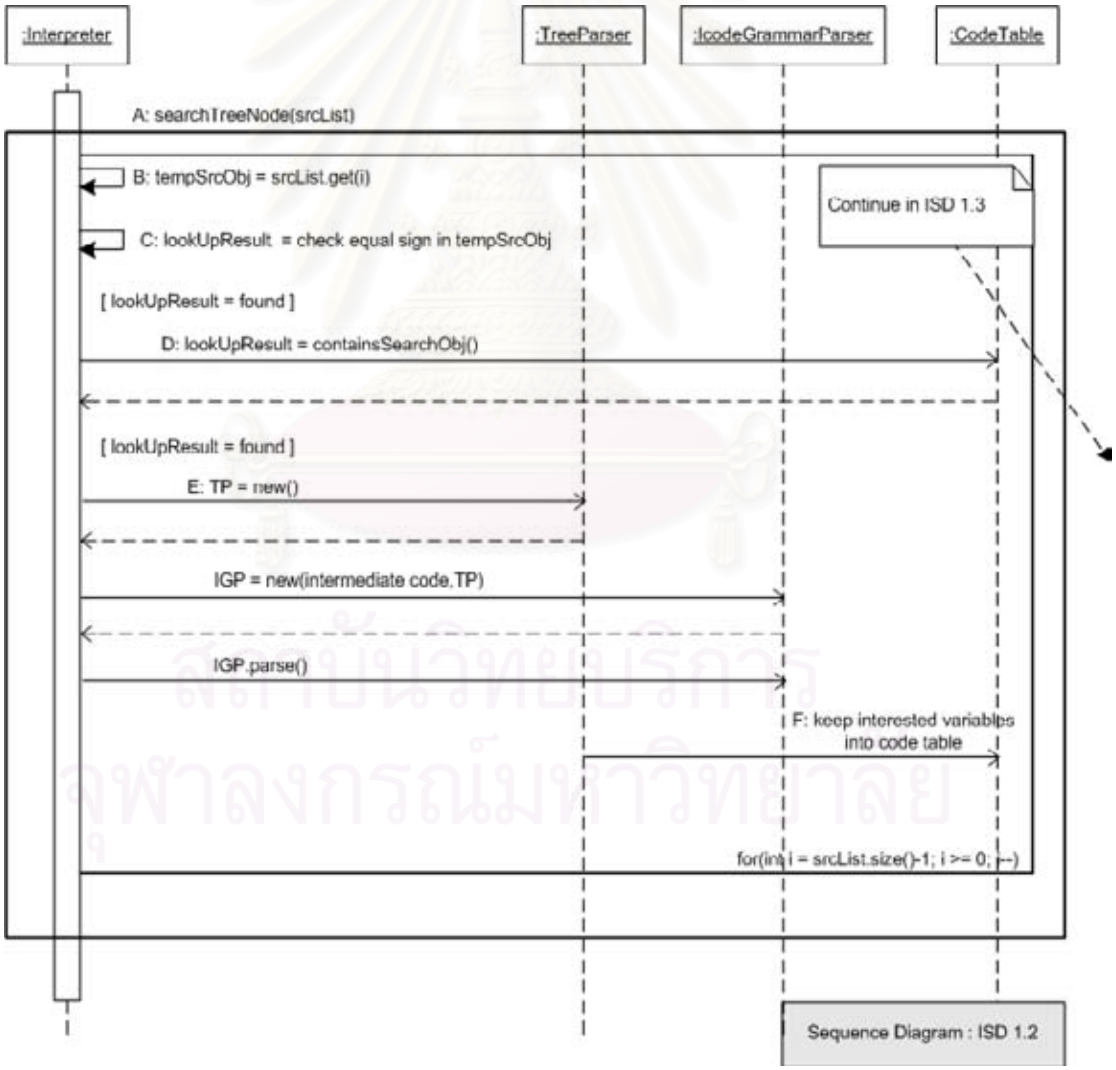


จากนั้นแบ่งคำสั่งอินเตอร์มิเดียทที่ได้แยกออกเป็นบรรทัดๆ โดยใช้อักขระพิเศษ “\n” เป็นตัวแบ่ง ขั้นตอนนี้ใช้คลาส StringTokenizer จากคลาสมาตราฐาน Sun’s JDK for

Java แบ่ง String โดยกำหนดตัวแบ่งเป็นอักขระพิเศษ “\n” คลาสนี้จะส่งคำสั่งอินเตอร์มิเดียทที่ถูกแบ่งออกเป็นรายการลำดับคำสั่ง

เมื่อได้ข้อมูลของแต่ละบรรทัด จากนั้นจะทำการแยกข้อมูลที่ต้องการออกมาโดยใช้เครื่องหมาย “;” และ “,” และส่งค่าที่แยกออกมาสร้าง InstructionSourceItem หนึ่งคำสั่งต่อหนึ่งอ็อบเจกต์ วนซ้ำจนครบทุกบรรทัดของคำสั่งอินเตอร์มิเดียท เพื่อสร้างรายการอ็อบเจกต์ InstructionSourceItem

1.3 Interpreter เรียกใช้เมทธอด initialTreeMap () สร้าง TreeNode เพื่อเก็บ OutputActor กำหนดให้เป็น TreeNode เริ่มต้น และเก็บชื่อของตัวแปรผลลัพธ์ไว้ใน CodeTable โดยใช้ชื่อของตัวแปรเป็นคำหลักในการค้นหา และค่าที่ใช้จัดเก็บคือ OutputActor ดังรูปที่ 4.28



รูปที่ 4.28 (ก) แผนภาพซีเควนซ์แสดงการทำงานส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์

1.4 จากนั้นค้นหาตัวแปรที่สนใจ (Searching interested variables) ในที่นี้ก็คือการค้นหาตัวแปรและคำสั่งที่มีความสัมพันธ์กับตัวแปรนำเข้าและตัวแปรผลลัพธ์ เพื่อนำมาสร้างโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์ โดยตัวแปรผลลัพธ์จะถือว่าเป็นตัวแปรที่สนใจตั้งต้นในการค้นหา ขั้นตอนนี้จะอยู่ในเมทธอด searchTreeNode () ดังแสดงในแผนภาพซีเควนซ์รูปที่ 4.28 (ก) กระบวนการทำงานมี ดังนี้

- Interpreter เรียกใช้เมทธอด searchTreeNode () รับรายการลำดับของ InstructionSourceItem เป็นพารามิเตอร์
- ภายในเมทธอด searchTreeNode () วนซ้ำดึงค่าคำสั่งอินเตอร์มิเดียทที่เก็บอยู่ภายใน InstructionSourceItem นำออกมาตรวจสอบ
- ค้นหาเครื่องหมาย “=” ซึ่งแสดงถึงความสัมพันธ์ของตัวแปร ระหว่างตัวแปรที่อยู่ทางซ้ายกับตัวแปรที่อยู่ทางขวาภายในคำสั่งอินเตอร์มิเดียท
- ถ้าพบ จะตรวจสอบต่อไปว่าด้านซ้ายของเครื่องหมาย “=” เป็นตัวแปรที่สนใจหรือไม่ โดยนำชื่อตัวแปรไปค้นหากับชื่อตัวแปรที่สนใจใน CodeTable
- ถ้าพบว่ามีตัวแปรที่สนใจภายในคำสั่งอินเตอร์มิเดียทก็จะส่งคำสั่งนั้นให้ TreeParser นำไปประมวลผล โดยมีกระบวนการทำงาน ดังนี้

1) นำชื่อตัวแปรที่อยู่ด้านซ้ายของเครื่องหมาย “=” ออกมาใช้ซึ่งก็คือตัวแปรที่สนใจซึ่งผ่านการสร้างอ็อบเจกต์แอคเตอร์จากบรรทัดคำสั่งก่อนหน้านี้ ยกเว้นตัวแปรผลลัพธ์ซึ่งจะถูกสร้างตั้งแต่เริ่มต้นการทำงาน โดยนำชื่อที่ได้เรียกใช้อ็อบเจกต์แอคเตอร์ ที่ถูกเก็บไว้ใน CodeTable ออกมาใช้งาน

2) ตรวจสอบด้านขวาของเครื่องหมาย “=” ว่าเป็นคำสั่งอินเตอร์มิเดียทสำหรับกำหนดค่าให้ตัวแปรด้านซ้ายมือชนิดใด ดังที่กำหนดไว้ในไวยากรณ์ภาษา ยกตัวอย่างเช่น

2.1) ถ้าพบว่าเป็นบรรทัดตรงกับการบรรยายไวยากรณ์ของ GETTER ที่นิยามไว้ประกอบด้วย digitget และ ชื่อตัวแปรนำเข้า เช่น xx = digitget x จะนำมาสร้างโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์ ดังนี้

- สร้างอ็อบเจกต์แอคเตอร์ฟังก์ชัน digitget โดยกำหนด actor property เป็น FunctionActor
- สร้างอ็อบเจกต์แอคเตอร์ตัวแปรนำเข้า โดยกำหนด actor property เป็น InputActor

- กำหนดให้อ็อบเจกต์แอคเตอร์ตัวแปรนำเข้า เป็น TreeNode ลูก ของอ็อบเจกต์แอคเตอร์ฟังก์ชัน digitget
- กำหนดให้อ็อบเจกต์แอคเตอร์ฟังก์ชัน digitget เป็น TreeNode ลูก ของอ็อบเจกต์แอคเตอร์ตัวแปรที่อยู่ด้านซ้ายของเครื่องหมาย “=”

2.2) ถ้าพบว่าเป็นบรรทัดการบรรยายไวยกรณ์ OPERATOR จะแบ่งออกมาได้เป็น 2 กรณีคือ

- กรณีที่ OPERATOR มีโทเค็นเดียว จะทำการสร้างอ็อบเจกต์แอคเตอร์ของโทเค็นนั้น ซึ่งอาจจะเป็นค่าคงที่หรือตัวแปรก็ได้ จากนั้นจึงกำหนดให้ อ็อบเจกต์แอคเตอร์ที่สร้างขึ้นมาเป็น TreeNode ลูก ของอ็อบเจกต์แอคเตอร์ที่อยู่ทางด้านซ้ายของเครื่องหมาย “=”
- กรณีที่ OPERATOR มี 3 โทเค็น ตัวอย่างเช่น A + B จะทำการสร้างอ็อบเจกต์แอคเตอร์ ดังนี้

1.สร้างอ็อบเจกต์แอคเตอร์ตัวถูกดำเนินการตัวที่ 1 โดยกำหนดตามประเภทของตัวถูกดำเนินการนั้น ซึ่งอาจจะเป็นค่าคงที่หรือตัวแปร

2.สร้างอ็อบเจกต์แอคเตอร์ตัวถูกดำเนินการตัวที่ 2 โดยกำหนดตามประเภทของตัวถูกดำเนินการนั้นซึ่งอาจจะเป็นค่าคงที่หรือตัวแปร

3.สร้างอ็อบเจกต์แอคเตอร์ตัวดำเนินการ โดยกำหนดแอคเตอร์ ตามประเภทของตัวดำเนินการนั้น

4.กำหนดให้อ็อบเจกต์แอคเตอร์ตัวถูกดำเนินการตัวที่ 1 และ 2 เป็น TreeNode ลูก ของอ็อบเจกต์แอคเตอร์ตัวดำเนินการ

5.กำหนดให้อ็อบเจกต์แอคเตอร์ตัวดำเนินการเป็น TreeNode ลูก ของอ็อบเจกต์แอคเตอร์ตัวแปรด้านซ้ายของเครื่องหมายเท่ากับ

- กรณีที่มีเครื่องหมายอยู่ติดกับค่าคงที่เช่น “a-1” หรือ “2-x” จะทำการรวมเครื่องหมายและค่าคงที่เข้าไปด้วยกัน แล้วสร้าง TreeNode ที่รวมทั้งเครื่องหมายและค่าคงที่นั้น จากนั้นจึงนำไปเชื่อมต่อกับ TreeNode ตัวแปรอีกตัวที่เหลืออยู่ โดยชื่อของ TreeNode จะกลายเป็น “-1” หรือ “2-”
- กรณีที่โทเค็นมีเครื่องหมาย ลบ (-) , เครื่องหมายนิเสธ (not) หรือเครื่องหมายนิเสธของ bitstring (~) จะสร้าง TreeNode เพิ่มเข้ามาเชื่อมต่อกับ TreeNode ของตัวถูกกระทำนั้น เช่น “–A” จะมีการสร้างแอคเตอร์ “A” จากนั้นจึงสร้างแอคเตอร์ “-” มาเป็น TreeNode ลูกของ TreeNode “A”

2.3) ถ้าพบว่าเป็นบรรทัดของการบรรยายไวยากรณ์ RETURNPARAMOBJ ซึ่งเป็นบรรทัดที่รับค่าส่งคืนมาจากฟังก์ชันที่ผู้ใช้งานสร้างขึ้น จะมีการค้นหาชื่อฟังก์ชันและพารามิเตอร์ก่อนที่จะส่งคำสั่งอินเตอร์มิเดียทให้กับคลาส TreeParser โดยจะนำสร้างความสัมพันธ์ดังนี้

- สร้างอ็อบเจกต์แอคเตอร์ฟังก์ชัน
- สร้างอ็อบเจกต์แอคเตอร์พารามิเตอร์ ถ้าพบการประกาศพารามิเตอร์
- กำหนดให้แอคเตอร์พารามิเตอร์เป็น TreeNode ลูก ของแอคเตอร์ฟังก์ชัน
- กำหนดให้อ็อบเจกต์แอคเตอร์ของฟังก์ชันเป็น TreeNode ลูก ของอ็อบเจกต์แอคเตอร์ตัวแปรที่อยู่ด้านซ้ายของเครื่องหมาย “=”
- เก็บอ็อบเจกต์แอคเตอร์ตัวใหม่เข้าไปใน CodeTable และเก็บชื่อของพารามิเตอร์ไว้ในรายการลำดับของตัวแปรที่สนใจใน CodeTable

2.4) ถ้าพบเป็นบรรทัดของการเรียกใช้ฟังก์ชันภายใน เช่น inttobitstr (), bitstrtoint () เป็นต้น จะทำการสร้างความสัมพันธ์ได้ดังนี้

- สร้างอ็อบเจกต์แอคเตอร์พารามิเตอร์แต่ละฟังก์ชัน
- สร้างอ็อบเจกต์แอคเตอร์ฟังก์ชัน โดยกำหนด actor property เป็น FunctionActor
- กำหนดให้อ็อบเจกต์แอคเตอร์พารามิเตอร์เป็น TreeNode ลูกของอ็อบเจกต์แอคเตอร์ฟังก์ชัน
- กำหนดให้อ็อบเจกต์แอคเตอร์ฟังก์ชันเป็น TreeNode ลูกของอ็อบเจกต์แอคเตอร์ตัวแปรด้านซ้ายของเครื่องหมาย “=”
- เก็บชื่อพารามิเตอร์เข้าไปไว้ในรายการลำดับของตัวแปรที่สนใจ และเก็บอ็อบเจกต์แอคเตอร์ที่สร้างใหม่ไว้ใน CodeTable

3) การค้นหาจุด Condition มีไว้เพื่อเป็นการแบ่งชั้นของความสัมพันธ์ตามเงื่อนไข (Condition) ในคำสั่งอินเตอร์มิเดียท เพื่อบอกจุดกระโดดข้ามและจุดเชื่อมต่อของคำสั่ง โดยจุดจะแบ่งออกเป็น 2 ชนิดคือ start point และ end point

- จุดชนิด start point คือจุดที่อยู่ระหว่างคำสั่งอินเตอร์มิเดียทก่อนหน้าคำสั่งทางเลือกใดๆ เป็นตัวบอกจุดเริ่มต้นของประโยคคำสั่งที่อยู่ภายใต้เงื่อนไข

- จุดชนิด end point คือจุดที่อยู่ระหว่างตัวแปรชั่วคราวที่เก็บค่าตรรกะของประโยคเงื่อนไขกับคำสั่งอินเตอร์มิเดียทที่อยู่ภายใน condition เพื่อแยกระหว่างคำสั่งอินเตอร์มิเดียทที่เป็นเงื่อนไขกับคำสั่งอินเตอร์มิเดียทที่อยู่ภายใต้เงื่อนไขนั้นๆ

วิธีการขั้นตอนของการสร้างจุด start point และ end point มีดังนี้

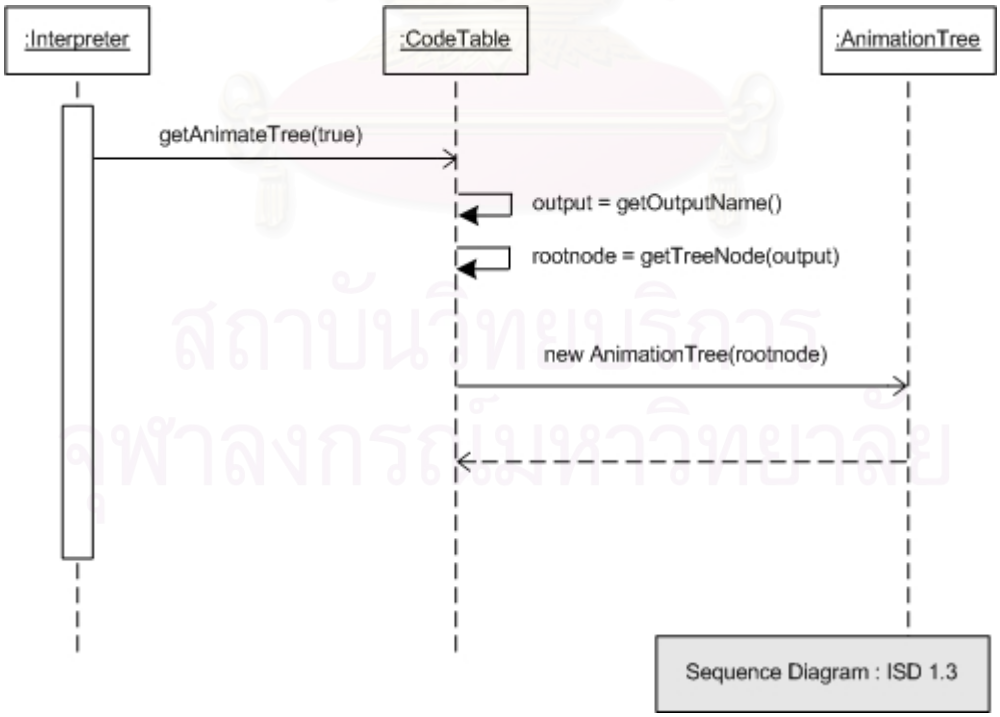
- ระหว่างการค้นหาเครื่องหมาย “=” ในแต่ละบรรทัดของคำสั่งอินเตอร์มิเดียทเพื่อสร้างโครงสร้างข้อมูลสำหรับแสดงภาพมโนทัศน์ จะมีการตรวจสอบเพิ่มเติมว่าบรรทัดของคำสั่งมี “if” เป็นคำเริ่มต้นหรือไม่
- ถ้าไม่พบ “if” เป็นคำเริ่มต้น และบรรทัดนั้นมีเครื่องหมาย “=” และค่าระดับตัวแปรมากกว่า 0 ซึ่งหมายความว่าบรรทัดนั้นอยู่ภายใต้ประโยคเงื่อนไขใดเงื่อนไขหนึ่งและค่าของตัวแปรตรวจสอบ “startCondition” เป็นเท็จ ซึ่งหมายความว่าคำสั่งอินเตอร์มิเดียทบรรทัดนี้ยังไม่ได้ผ่านบรรทัดที่ใช้เป็นประโยคเงื่อนไข และยังอยู่ภายใน เงื่อนไขนั้นๆ ให้นำอ็อบเจกต์แอคเตอร์ที่อยู่ทางขวาของเครื่องหมาย “=” ไปเก็บไว้ในรายการของตัวแปรที่จะใช้ต่อกับอ็อบเจกต์แอคเตอร์จุด โดยก่อนที่จะเก็บค่าเข้าไปในรายการของตัวแปร ต้องทำการล้างค่างเก่าที่อยู่ภายในรายการของตัวแปรทิ้งไปก่อน เพราะต้องการค่า TreeNode ของบรรทัดที่สุดท้ายที่อยู่ติดกับบรรทัดเงื่อนไขเท่านั้น
- ถ้าไม่พบ “if” เป็นคำเริ่มต้นและค่าของตัวแปร “startCondition” มีค่าเท่ากับ “True” และมีค่าระดับตัวแปรตรงกับระดับตัวแปรของบรรทัดที่เป็นเงื่อนไขแสดงว่าบรรทัดนี้เป็นบรรทัดของเงื่อนไข ก็จะสร้างอ็อบเจกต์แอคเตอร์ นำไปกำหนดให้เป็นโหนดลูกของจุด end point และเก็บอ็อบเจกต์แอคเตอร์ ที่สร้างขึ้นมาใหม่ไว้ในรายการตัวแปรเพื่อให้สามารถนำไปต่อกับจุด start point ต่อไปได้
- ถ้าไม่พบ “if” เป็นคำเริ่มต้น และค่า “startCondition” เป็นจริงและค่าระดับตัวแปรมีค่าน้อยกว่าค่าระดับของจุด แสดงว่าบรรทัดนี้เป็นบรรทัดคำสั่งอินเตอร์มิเดียทของเงื่อนไขอื่น จากนั้นจะทำการสร้างอ็อบเจกต์แอคเตอร์จุดขึ้นมาใหม่และกำหนดให้เป็นชนิด start ซึ่งเป็น TreeNode ลูก ของ TreeNode ที่เก็บอยู่ภายในไว้ในลิสต์
- ถ้าพบโทเค็น “if” เป็นคำเริ่มต้นแสดงว่าบรรทัดนั้นเป็นบรรทัดเงื่อนไข ซึ่งมีข้อมูลของชื่อตัวแปรชั่วคราวเก็บค่าตรรกะที่ใช้ในเงื่อนไขนี้และจุดกระโดด ที่ใช้ในการกระโดดของคำสั่งอินเตอร์มิเดียท ก็จะทำการสร้างอ็อบเจกต์แอคเตอร์จุด โดยกำหนดชื่อจุดให้เป็นชื่อของตัวแปรชั่วคราว, กำหนดแอคเตอร์เป็น PointActor และกำหนดชนิดของจุดให้เป็น end point เมื่อสร้างเสร็จจึงดึงค่า TreeNode ที่เก็บไว้ในรายการเพื่อทำการสร้างจุดออกมา แล้ว

กำหนด TreeNode ที่ดึงค่าออกมาให้เป็นโหนดแม่ของจุด end point ที่สร้างขึ้นมาใหม่และกำหนดค่า “startCondition” ให้เป็น “True”

ทุกบรรทัดที่ถูกส่งไปให้ TreeParser จะต้องทำการเก็บบรรทัดเหล่านั้นไว้ใน CodeTable เพื่อใช้ตรวจสอบว่าบรรทัดของคำสั่งอินเตอร์มิเดียทที่จะนำไปสร้างคำสั่งสำหรับการแสดงภาพมโนทัศน์ในขั้นตอนถัดไป

เมื่อกำหนดความสัมพันธ์เสร็จเรียบร้อยแล้ว จึงนำอ็อบเจกต์แอคเตอร์ที่สร้างไว้ทั้งหมด เข้าไปเก็บไว้ใน CodeTable รวมทั้งเก็บชื่อตัวแปรที่ถูกนำมาสร้างเป็นอ็อบเจกต์แอคเตอร์เข้าไปในลิสต์ของตัวแปรที่สนใจเพื่อใช้ในการค้นหาตัวแปรที่สนใจตัวอื่นต่อไป

จากนั้นวนซ้ำนำรายการลำดับคำสั่งอินเตอร์มิเดียทจนครบ เพื่อสร้างอ็อบเจกต์แอคเตอร์ทั้งหมดเก็บไว้ใน CodeTable โดยอ็อบเจกต์แอคเตอร์แต่ละตัวจะมีการระบุความสัมพันธ์ระหว่างกันไว้แล้วว่า อ็อบเจกต์แอคเตอร์มี TreeNode ลูก หรือ TreeNode แม่เป็นอ็อบเจกต์แอคเตอร์ตัวใด จากนั้นจึงส่งอ็อบเจกต์แอคเตอร์ของตัวแปรผลลัพธ์ซึ่งเป็น TreeNode รากของอ็อบเจกต์แอคเตอร์ทุกตัวใช้เป็นพารามิเตอร์สร้าง AnimationTree ซึ่งจะถูกเรียกใช้งานจากส่วนประมวลผลสร้างภาพมโนทัศน์ในขั้นตอนการวาดรูปโครงสร้างความสัมพันธ์ต่อไป ดังแผนภาพซีเควนซ์ ดังรูปที่ 4.28 (ข)



รูปที่ 4.28 (ข) แผนภาพซีเควนซ์แสดงการทำงานส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์

ตัวอย่าง การค้นหาตัวแปรที่สนใจภายในคำสั่งอินเตอร์มิเดียท

```
main_integer_a = 4;
main_integer_b = 7;
$t4 = main_integer_a * main_integer_b
$t5 = main_integer_a + main_integer_b
z = $t5
```

จากตัวอย่าง ตัวแปร z คือตัวแปรผลลัพธ์และจะถูกสร้างเป็นอ็อบเจกต์แอคเตอร์เริ่มต้นและถูกเก็บชื่อไว้ในลำดับรายการตัวแปรที่สนใจ ขั้นตอนจะเริ่มทำการค้นหา จากบรรทัดล่างขึ้นบน ดังนี้

1) จะตรวจสอบตัวแปรใดๆ ที่กำหนดค่าให้กับตัวแปรผลลัพธ์จากตัวอย่างนี้ z = $t5 จะพบว่าเป็นบรรทัดที่มีเครื่องหมาย “=” จากนั้นเมื่อตรวจสอบตัวแปรด้านซ้ายก็จะพบว่าอยู่ในลำดับรายการของตัวแปรที่สนใจ ดังนั้นจึงส่งคำสั่งนี้ให้ TreeParser โดยตัดโทเค็นของคำสั่งแยกตัวแปรที่สนใจคือ $5 นำมาสร้างเป็นอ็อบเจกต์แอคเตอร์และเก็บชื่อไว้ในลำดับรายการของตัวแปรที่สนใจ

2) จากนั้นค้นหาบรรทัดถัดไปคือ $t5 = main_integer_a + main_integer_b ตัวแปร $t5 พบว่าเป็นตัวแปรที่สนใจจากขั้นตอนก่อนหน้าจึงส่งคำสั่งบรรทัดนี้ให้ TreeParser สร้างเป็นอ็อบเจกต์แอคเตอร์ของ main_integer_a และ main_integer_b และเก็บชื่อไว้ในลำดับรายการของตัวแปรที่สนใจ

3) บรรทัดถัดไปคือ $t4 = main_integer_a * main_integer_b บรรทัดนี้จะไม่ตรงตามเงื่อนไข เพราะ $t4 ไม่ใช้ตัวแปรที่สนใจทำให้ไม่ส่งบรรทัดนี้ไปหาคลาส TreeParser บรรทัดที่เหลือก็จะถูกส่งไปให้ TreeParser ทั้ง 2 บรรทัด เพราะ main_integer_a และ main_integer_b ซึ่งเป็นตัวแปรที่สนใจทั้งคู่ แต่สำหรับค่าคงที่ 4 และ 7 จะไม่ถูกเก็บเข้าเป็นตัวแปรที่สนใจเพราะไม่ใช่ตัวแปร

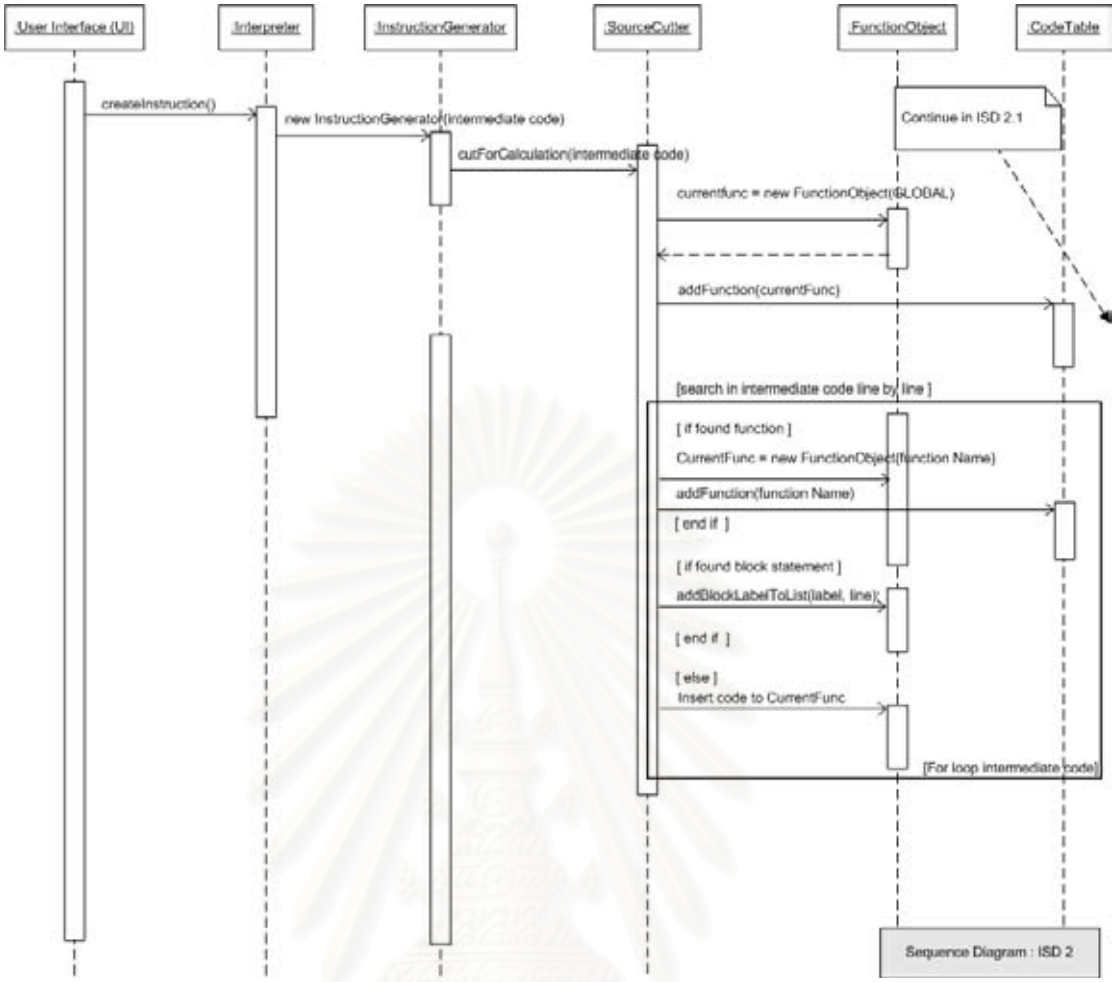
2. ขั้นตอนการแปลงชุดคำสั่งอินเตอร์มิเดียทเป็นคำสั่งสำหรับแสดงภาพมโนทัศน์

เป็นขั้นตอนเกิดขึ้นต่อเนื่องจากขั้นตอนการสร้างโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์เสร็จเรียบร้อยแล้ว ดังรูปที่ 4.29 การทำงานเริ่มจากผู้ออกแบบอัลกอริทึมสั่งคำสั่ง Start เริ่มการทำงานผ่านทางส่วนประสานงานผู้ใช้ โดยมีลำดับกิจกรรม ดังนี้

1. Interpreter เรียกใช้เมทธอด createInstruction () เพื่อสร้าง Instruction Generator

2. InstructionGenerator ทำหน้าที่ส่งคำสั่งอินเตอร์มิเดียทให้ SourceCutter ตัดแบ่งโค้ดคำสั่งอินเตอร์มิเดียทแยกตามฟังก์ชัน และคำสั่งภายในฟังก์ชันนั้นๆ ผ่านทางเมทธอด cutCodeForCalculation () เพื่อแบ่งคำสั่งอินเตอร์มิเดียทออกเป็นบรรทัดๆ จากนั้นสร้าง FunctionObject เริ่มต้นโดยกำหนดให้ชื่อฟังก์ชันเป็น “global” และกำหนดให้เป็นฟังก์ชันปัจจุบัน จากนั้นส่งคำสั่งอินเตอร์มิเดียทแต่ละบรรทัดจัดเก็บในฟังก์ชันเริ่มต้น โดยมีเงื่อนไขดังนี้

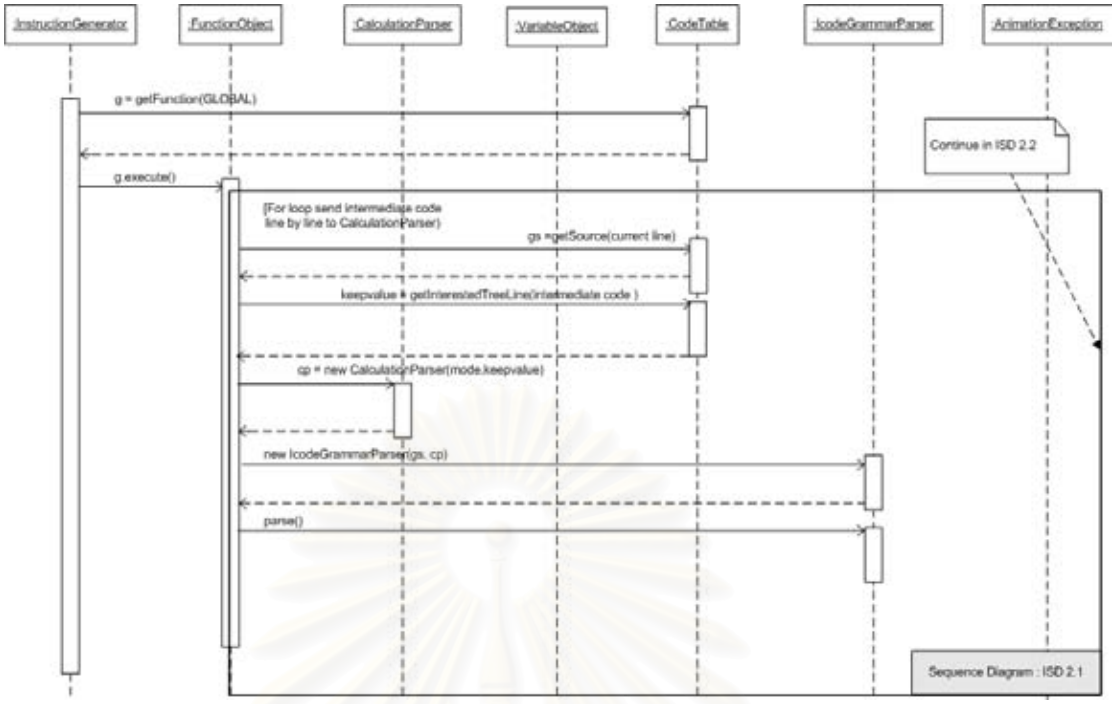
- ถ้าเป็นบรรทัดคำสั่งอินเตอร์มิเดียทที่ขึ้นต้นด้วยชื่อฟังก์ชัน เช่น “CACULLATE:” จะทำการสร้าง FunctionObject ขึ้นมาใหม่ และจัดเก็บไว้ใน CodeTable จากนั้นกำหนดการอ้างอิงการทำงานของฟังก์ชันนั้นให้เป็นฟังก์ชันปัจจุบัน ถ้าฟังก์ชันนั้นเป็นฟังก์ชัน “main” จะส่งคำสั่งอินเตอร์มิเดียท “main:” ไปเก็บไว้ในฟังก์ชัน global จากนั้นสร้าง FunctionObject และกำหนดฟังก์ชัน “main” เป็นฟังก์ชันปัจจุบัน เพื่อหาจุดเริ่มต้นของการทำงาน
- ถ้าเป็นบรรทัดที่ขึ้นต้นด้วยชื่อบล็อก เช่น “@2:” จะส่งเข้าไปเก็บเป็นชื่อบล็อกใหม่ในอ็อบเจกต์ฟังก์ชันปัจจุบัน พร้อมทั้งระบุหมายเลขบรรทัดของบล็อกนี้ เพื่อที่จะสามารถทำการกระโดดข้ามบรรทัดได้ ในกรณีตรวจพบการใช้คำสั่ง “goto”
- กรณีอื่นๆ จะส่งคำสั่งอินเตอร์มิเดียทไปเก็บไว้ในฟังก์ชันปัจจุบันตามปกติ



รูปที่ 4.29 แผนภาพซีเควนซ์การแปลงชุดคำสั่งอินเตอร์มิเดียทเป็นคำสั่งสร้างภาพมโนทัศน์

การนำคำสั่งอินเตอร์มิเดียทไปเก็บไว้ในอ็อบเจกต์ FunctionObject จะต้องแปลงคำสั่งอินเตอร์มิเดียทเป็นอ็อบเจกต์ InstructionSourceItem เพื่อเรียกใช้ข้อมูลของแต่ละบรรทัดของคำสั่งอินเตอร์มิเดียทได้ง่ายขึ้น เมื่อแปลงเสร็จจะถูกส่งเข้ามาเก็บไว้ใน FunctionObject ในรูปแบบของรายการทีละลำดับ

เมื่อแยกภาษาอินเตอร์มิเดียททั้งหมดเรียบร้อยแล้ว จะได้อ็อบเจกต์ FunctionObject ของแต่ละฟังก์ชันในคำสั่งอินเตอร์มิเดียท อย่างน้อยจะต้องมีฟังก์ชัน “global” และ “main” โดยฟังก์ชันทั้งหมดจะถูกเก็บไว้ใน CodeTable โดยใช้คำหลักคือชื่อฟังก์ชัน และค่าที่จัดเก็บคือ FunctionObject



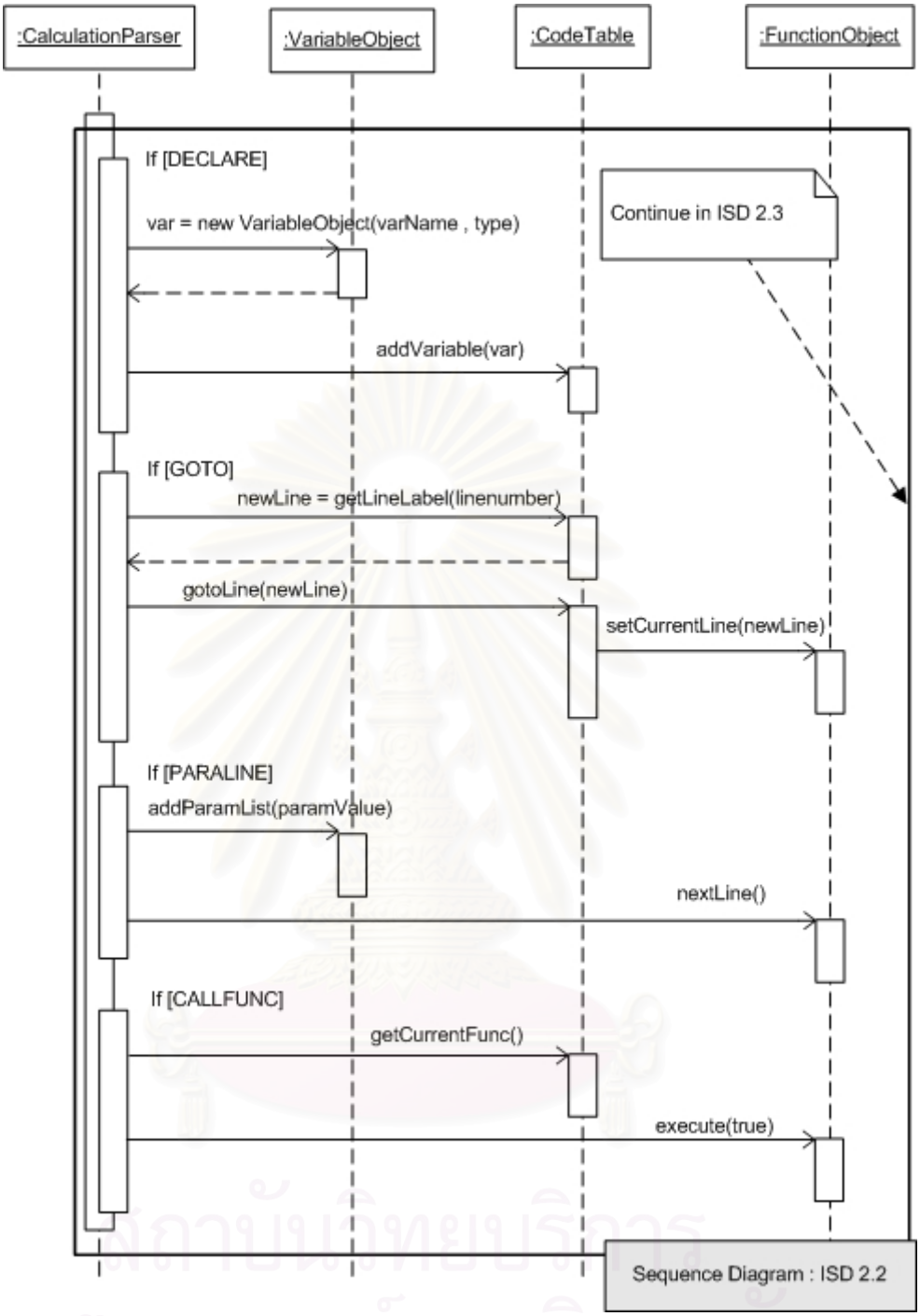
รูปที่ 4.29 (ก) แผนภาพซีเควนซ์การแปลงชุดคำสั่งอินเตอร์มิเดียทเป็นคำสั่งสำหรับสร้างภาพมโนทัศน์

3. InstructionGenerator เรียกใช้เมทธอด execute () ของฟังก์ชัน global โดยส่งคำสั่งอินเตอร์มิเดียทให้ CalculationParser คำนวณคำสั่งอินเตอร์มิเดียทตามรูปแบบที่กำหนดไว้ในการบรรยายไวยากรณ์ โดยมีขั้นตอนการทำงานขึ้นกับการบรรยายไวยากรณ์ ดังนี้

- ถ้าพบบรรทัดเป็นการบรรยาย “DECLARE” คือบรรทัดการประกาศตัวแปร จะสร้าง VariableObject สำหรับใช้เก็บข้อมูลตัวแปร ไว้ใน CodeTable โดยใช้คำหลักคือ ชื่อตัวแปร และค่าที่จัดเก็บคือ VariableObject

- ถ้าพบบรรทัดเป็นการบรรยาย “GOTO” จะกระโดดข้ามไปยังบล็อกของฟังก์ชันนั้นๆ โดยการดึงค่าหมายเลขบรรทัดของบล็อกมาจากตารางแฮช ซึ่งจัดเก็บอยู่ภายในอ็อบเจกต์ฟังก์ชัน เมื่อได้ค่าของบรรทัดจึงกำหนดค่าบรรทัดนี้เป็นค่าบรรทัดปัจจุบัน เพื่อที่จะใช้เป็นบรรทัดถัดไปในการคำนวณ

- ถ้าพบบรรทัดเป็นการบรรยาย “PARAMLINE” คือบรรทัดที่บอกว่าพารามิเตอร์ของฟังก์ชันถัดไปคืออะไร พารามิเตอร์ของฟังก์ชันสามารถมีได้ไม่จำกัด ดังนั้นจึงจำเป็นต้องมีการเก็บไว้ภายใน CodeTable โดยจะดึงค่าที่เป็นพารามิเตอร์เข้าไปเก็บไว้ชั่วคราว ในลักษณะของรายการลำดับ โดยจะเก็บเข้ารายการลำดับไปจนกว่าจะเจอบรรทัดที่เรียกฟังก์ชันนั้นขึ้นมาทำงาน



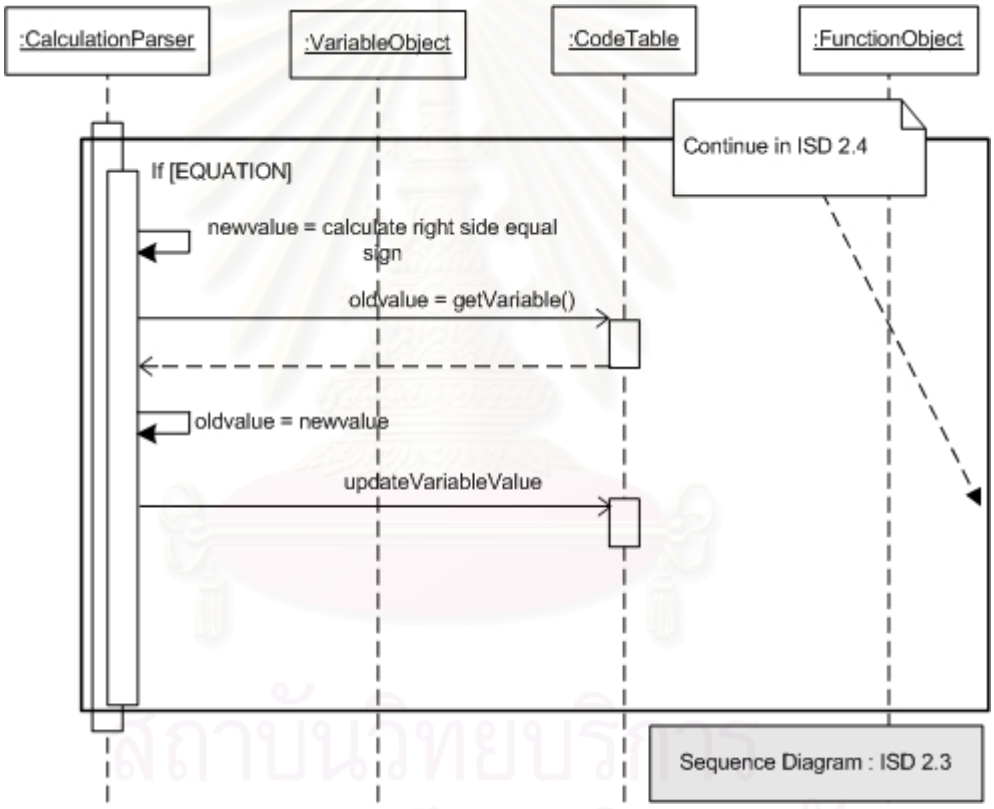
รูปที่ 4.29 (ข) แผนภาพซีเควนซ์การแปลงชุดคำสั่งอินเตอร์มิเดียทเป็นคำสั่งสำหรับสร้างภาพมโนทัศน์

- ถ้าพบบรรทัดเป็นการบรรยาย “CALLFUNC” จะทำการเรียกฟังก์ชันที่อยู่ในบรรทัดนี้ขึ้นมาทำงาน โดยการดึงค่า FunctionObject จาก CodeTable จากนั้นเรียกเมทธอด excecute () เพื่อเริ่มการทำงานของคำสั่งภายในฟังก์ชันนั้นๆ โดยภายในฟังก์ชันอาจมีการเรียกใช้

ค่าพารามิเตอร์ซึ่งได้ถูกเก็บไว้ในรายการภายใน CodeTable ก่อนหน้านี้ออกมา โดยภายในรายการจะเป็นชื่อของตัวแปรหรือค่าคงที่ซึ่งจะมาใช้ในการคำนวณ ดังนี้

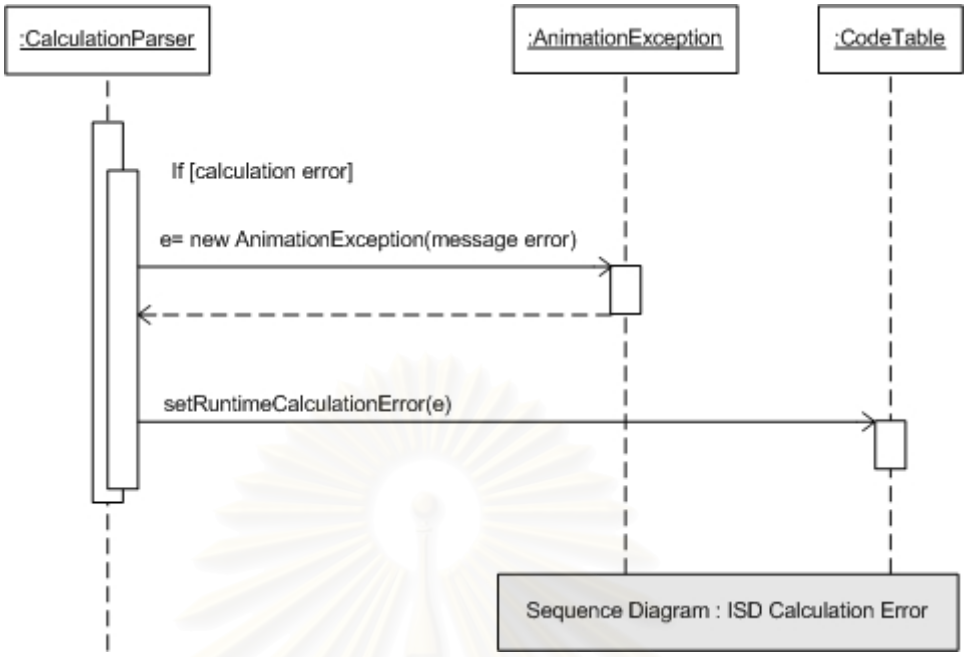
- ถ้าเป็นชื่อของตัวแปรจะถูกส่งไปเรียก VariableObject ออกมาจากคลาส CodeTable เพื่อที่จะได้ค่าของตัวแปรนี้
- ถ้าเป็นค่าคงที่ จะนำไปคำนวณหาผลลัพธ์

- ถ้าพบบรรทัดเป็นการบรรยาย “EQUATION” จะทำการคำนวณค่าทางด้านขวาของเครื่องหมาย “=” จากนั้นดึงค่า VariableObject ของตัวแปรด้านซ้าย ออกมาจาก CodeTable ที่ถูกสร้างตอนประกาศตัวแปร นำมาปรับปรุงค่าของตัวแปรจากนั้นจึงจัดเก็บกลับเข้าไปใน CodeTable ใหม่เพื่อรอเรียกใช้งานต่อไป ดังรูปที่ 4.29 (ค)

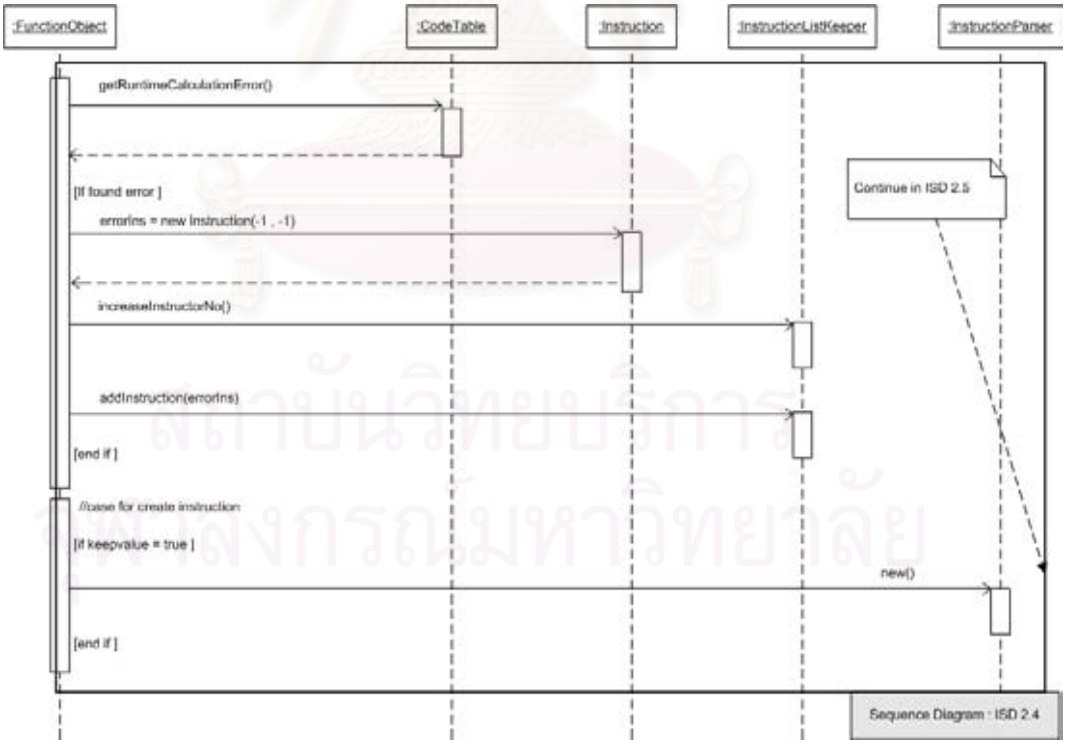


รูปที่ 4.29 (ค) แผนภาพซีเควนซ์การแปลงชุดคำสั่งอินเตอร์มิเดียทเป็นคำสั่งสำหรับสร้างภาพมโนทัศน์

การทำงานของ CalculationParser ถ้าพบข้อผิดพลาด (Runtime Error) จะทำการสร้างอ็อบเจกต์ AnimationException และจัดเก็บข้อผิดพลาดที่เกิดขึ้นซึ่งจะถูกนำไปสร้างคำสั่งสำหรับแสดงข้อผิดพลาดในระหว่างการแสดงภาพมโนทัศน์ เมื่อคำนวณค่าได้เสร็จแล้วในแต่ละบรรทัด จะรู้ว่า ณ ขณะนั้นตัวแปรที่อยู่ในบรรทัดมีค่าใดบ้าง เพื่อที่จะได้นำมาสร้างคำสั่งสำหรับสร้างภาพมโนทัศน์ในขั้นตอนถัดไป ดังรูปที่ 4.29 (ง)



รูปที่ 4.29 (ง) แผนภาพซีเควนซ์การแปลงชุดคำสั่งอินเตอร์มิเดียทเป็นคำสั่งสำหรับสร้างภาพมโนทัศน์



รูปที่ 4.29 (จ) แผนภาพซีเควนซ์การแปลงชุดคำสั่งอินเตอร์มิเดียทเป็นคำสั่งสำหรับสร้างภาพมโนทัศน์

4. FunctionObject ตรวจสอบบรรทัดที่ผ่านการคำนวณมาแล้วนั้นอยู่ในรายการของบรรทัดที่ถูกนำไปวาดภาพหรือไม่ ถ้าอยู่จึงทำการส่งไปให้ InstructionParser นำไปสร้างอ็อบเจกต์ Instruction ดังรูปที่ 4.29 (จ)

เมื่อทำการคำนวณค่าภายในบรรทัดเสร็จแล้ว จะรู้ว่ามีการเคลื่อนที่ของข้อมูลจากแอคเตอร์ใดไปที่แอคเตอร์ตัวใดบ้าง และรู้ว่าค่าที่เคลื่อนที่มีค่าเป็นอะไรเพราะฉะนั้นหลังจากคำนวณเสร็จแต่ละบรรทัด ก็จะทำการสร้างคำสั่งแสดงภาพมโนทัศน์ ซึ่งก็คืออ็อบเจกต์ Instruction โดยอ็อบเจกต์นี้จะเก็บข้อมูล 4 ชนิดคือ

- หมายเลขของคำสั่ง โดยค่านี้จะเป็นตัวกำหนดลำดับของการแสดงภาพมโนทัศน์ โดยจะเรียงลำดับการแสดงจากน้อยไปหามาก
- กำหนดฟิลด์ “rightSide” ของ Instruction ซึ่งก็คือแอคเตอร์ โดยฟิลด์นี้จะเป็นตัวบอกว่าแอคเตอร์ตัวไหนบนส่วนแสดงภาพมโนทัศน์ใช้เป็นตัวส่งค่า
- กำหนดฟิลด์ “leftSide” ของ Instruction ซึ่งก็คือแอคเตอร์ โดยฟิลด์นี้จะเป็นตัวบอกว่าแอคเตอร์ตัวไหนบนส่วนแสดงภาพมโนทัศน์จะใช้เป็นตัวรับค่า
- ค่าที่ถูกส่งผ่านระหว่างการแสดงภาพมโนทัศน์

คำสั่งอินเตอร์มิเดียทที่สนใจในการสร้างภาพมโนทัศน์ จะเป็นบรรทัดที่ถูกนำไปวาดเส้นความสัมพันธ์ในขั้นตอนการแปลโปรแกรมต้นฉบับ ดังนั้นบรรทัดเหล่านั้นจึงเป็นบรรทัดที่ตรงกับการบรรยายไวยากรณ์โปรดักชัน “EQUATION” โดยบรรทัดที่ใช้ในการสร้างภาพมโนทัศน์จะถูกเก็บไว้ตั้งแต่ตอนวาดรูปในรูปแบบของลำดับรายการ โดย InstructionParser จะทำการสร้าง Instruction ถ้าตรวจพบว่าออกจากโปรดักชัน “EQUATION” โดยแบ่งกรณีของโปรดักชันที่อยู่ทางขวาของเครื่องหมาย “=” ได้ดังนี้

- กรณีพบการบรรยาย “GETTER” บรรทัดนี้จะเป็นการดึงค่าของตัวแปรนำเข้าออกมา 1 ตำแหน่ง โดยการส่งข้อมูลจะแบ่งออกมาได้ 2 คำสั่ง ดังนี้

a. คำสั่งการส่งข้อมูลจากตัวแปรนำเข้ามาที่ฟังก์ชัน digitget

b. คำสั่งการส่งข้อมูลจากฟังก์ชัน digitget มาที่อ็อบเจกต์แอคเตอร์ตัวแปรชั่วคราวที่อยู่ทางด้านซ้ายของเครื่องหมาย “=”

- กรณีพบการบรรยาย “OPERATOR” จะเป็นบรรทัดที่ต้องดึงค่าของตัวแปรที่ใช้ในการคำนวณ เพื่อนำมาแสดงค่าระหว่างการแสดงภาพมโนทัศน์ โดยรูปแบบของคำสั่งที่ใช้จะมี 3 แบบ คือ

1) ด้านขวามีโทเค็นเดียวเช่น z = a จะสร้างคำสั่งขึ้นมา 1 คำสั่งเพื่อส่งค่าจากอ็อบเจกต์แอคเตอร์จากด้านขวาไปยังอ็อบเจกต์แอคเตอร์ตัวด้านซ้าย โดยจะกำหนดค่าฟิลด์ “leftSide” เป็นอ็อบเจกต์แอคเตอร์ตัวแปรทางซ้าย และ กำหนดค่า ฟิลด์ “rightSide” เป็นอ็อบเจกต์แอคเตอร์ตัวแปรทางขวา จากนั้นจึงกำหนดค่าที่จะส่งโดยนำค่า VariableObject ของตัวแปรทางซ้าย ที่เก็บอยู่ใน CodeTable เพราะค่าที่อยู่ใน VariableObject นี้จะเป็นค่าล่าสุดภายหลังจากผ่านการคำนวณ

2) ด้านขวามี 3 โทเค็นเช่น z = a + b จะสร้างคำสั่งดังนี้

- กำหนดฟิลด์ “leftSide” เป็นอ็อบเจกต์แอคเตอร์ตัวแปรทางซ้าย
- กำหนดฟิลด์ “rightSide” เป็นอ็อบเจกต์แอคเตอร์ตัวดำเนินการ
- กำหนดค่าที่จะส่ง 3 ค่าคือ

- ค่าที่จะออกมาจากตัวถูกดำเนินการตัวที่ 1 ซึ่งเป็นค่าชั่วคราวจาก CodeTable เก็บไว้ในระหว่างการคำนวณ

- ค่าที่จะออกมาจากตัวถูกดำเนินการตัวที่ 2 ซึ่งเป็นค่าชั่วคราวจาก CodeTable เก็บไว้ระหว่างการคำนวณ

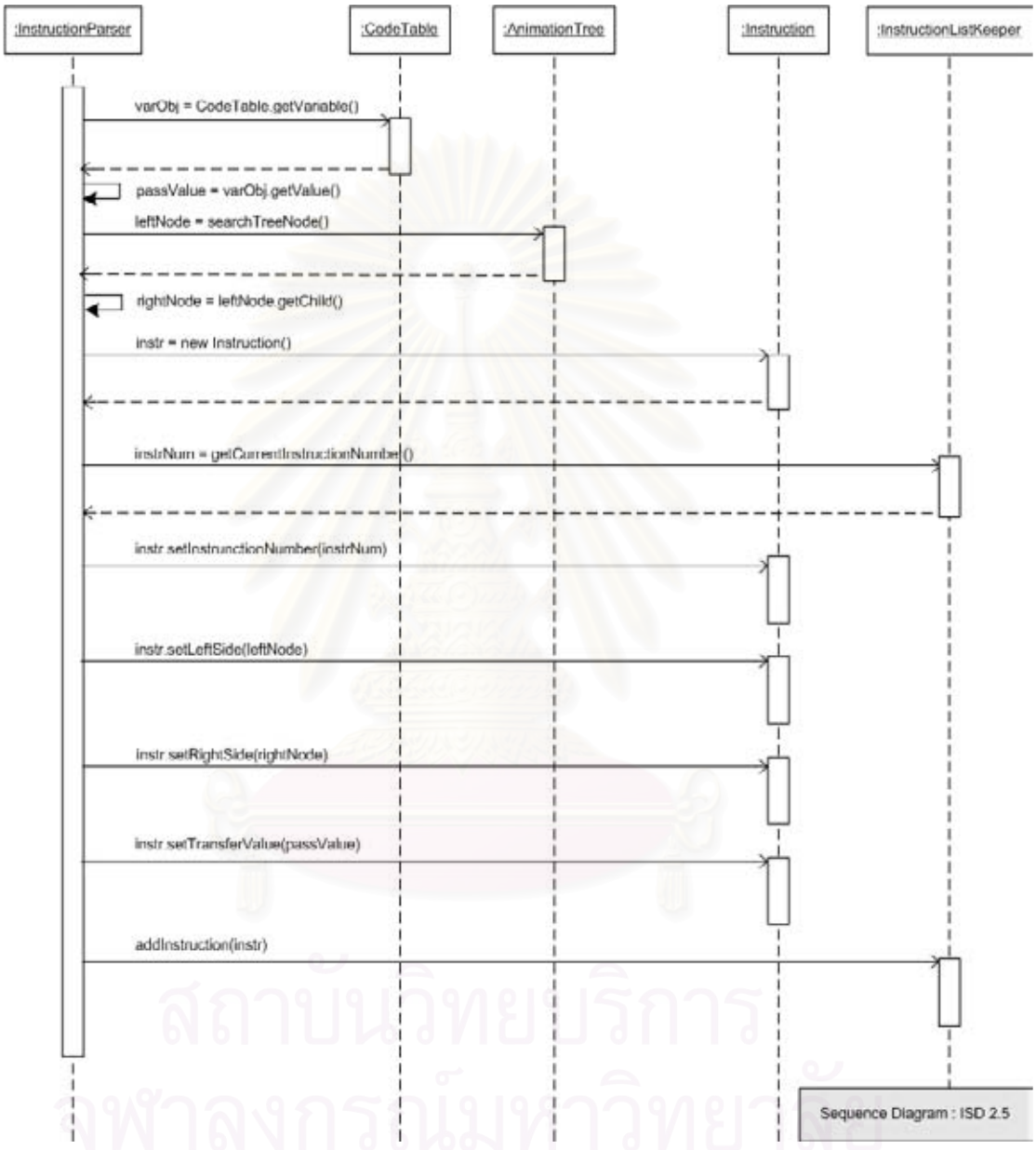
- ค่าที่จะออกมาจากตัวดำเนินการวิ่งไปหาตัวแปรทางซ้าย จะได้มาจากการดึงค่าจากอ็อบเจค VariableObject ของตัวแปรทางซ้ายที่อยู่ภายใน CodeTable

3) ด้านขวามีโทเค็นเครื่องหมาย “-” ,“~” หรือ “not” เช่น z = - a + b หรือ a + -b กรณีนี้จะเหมือนกับข้อด้านบนแต่จะเพิ่มคำสั่ง ขึ้นมาตามจำนวนเครื่องหมายที่เพิ่มขึ้นเพื่อที่จะเป็นการแสดงภาพมโนทัศน์ สำหรับการส่งข้อมูลจากตัวตัวถูกดำเนินการผ่านไปยังเครื่องหมายเหล่านั้นเพื่อแปลงค่าก่อนจะส่งไปถึงตัวดำเนินการ โดยจะสร้าง คำสั่ง ดังนี้

- กำหนดฟิลด์ “rightSide” เป็นอ็อบเจกต์แอคเตอร์ตัวถูกดำเนินการ
- กำหนดฟิลด์ “leftSide” เป็นเป็นอ็อบเจกต์แอคเตอร์ตัวเครื่องหมาย “-”, “~” หรือ “not”

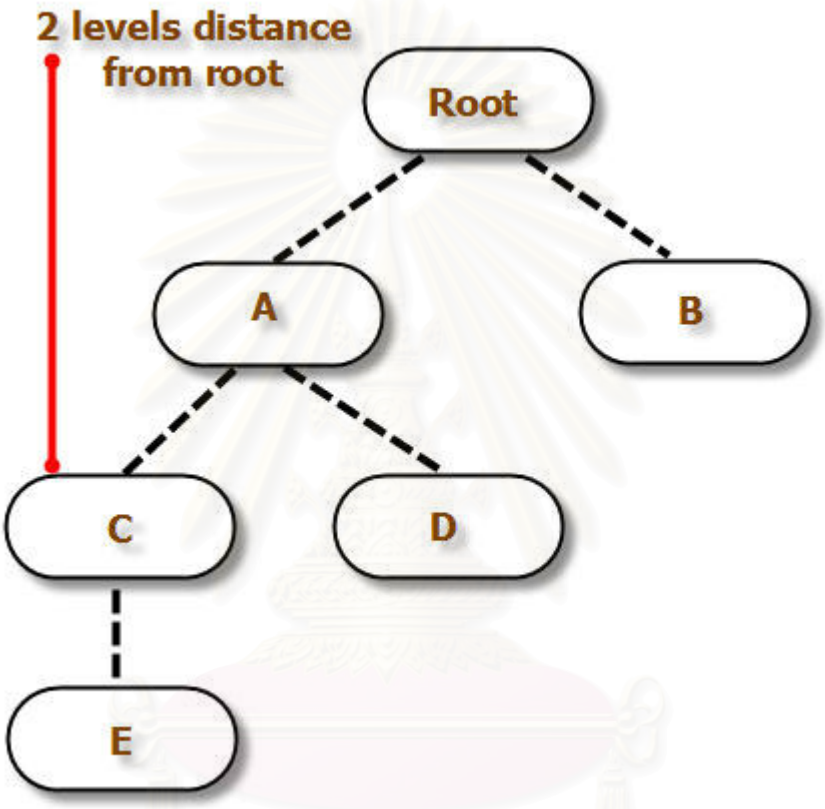
กำหนดค่าการส่งข้อมูลจากการดึงข้อมูลใน VariableObject ที่เก็บอยู่ใน CodeTable

- กรณีพบการบรรยายฟังก์ชันภายใน หรือฟังก์ชันที่ผู้ใช้สร้างขึ้นมา จะสร้างคำสั่งตามจำนวนพารามิเตอร์ เพื่อส่งค่าเข้ามาที่อ็อบเจกต์แอคเตอร์ฟังก์ชัน จากนั้นจึงส่งค่าจากฟังก์ชันไปที่อ็อบเจกต์แอคเตอร์ตัวแปรทางด้านซ้ายของเครื่องหมาย “=”



รูปที่ 4.29 (ฉ) แผนภาพซีเควนซ์การแปลงชุดคำสั่งอินเตอร์มิเดียทเป็นคำสั่งสำหรับสร้างภาพมโนทัศน์

InstructionParser จะส่งคำสั่งแสดงภาพมโนทัศน์นำไปจัดเก็บไว้ใน InstructionListKeeper ทุกครั้งหลังจากสร้างคำสั่งแสดงภาพมโนทัศน์เสร็จเรียบร้อยแล้ว เพื่อใช้เก็บอ็อบเจกต์คำสั่งที่ถูกสร้างขึ้นมาทั้งหมดและทำการกำหนดหมายเลขของคำสั่งทุกคำสั่ง เมื่อทำการสร้างครบทั้งหมดแล้วก็จะส่งคำสั่งแสดงภาพมโนทัศน์ทั้งหมด กำหนดให้กับ AnimationTree เพื่อนำไปใช้ในการสร้างภาพมโนทัศน์ต่อไป



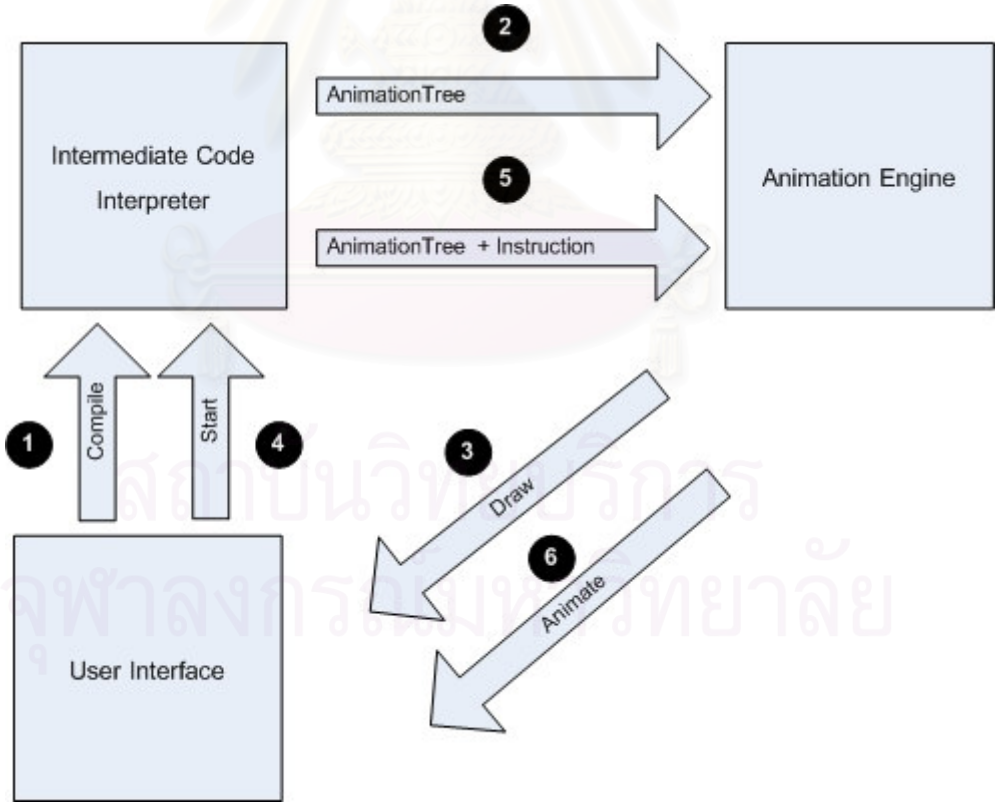
รูปที่ 4.30 โครงสร้างข้อมูลของ AnimationTree

4.2.2 ส่วนประมวลผลสร้างภาพมโนทัศน์ (Animation Engine)

การทำงานในส่วนการประมวลผลสร้างภาพมโนทัศน์ จะใช้โครงสร้างข้อมูล และชุดคำสั่งสำหรับสร้างภาพมโนทัศน์ จากส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ โดยแบ่งขั้นตอนการทำงานออกเป็น 2 ขั้นตอน คือ

1) ขั้นตอนการแสดงโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์ เกิดขึ้นเมื่อผู้ใช้สั่งการคอมไพล์โปรแกรมต้นฉบับ ส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ทำการสร้างโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์ หรือ AnimationTree ซึ่งใช้เก็บการอ้างอิงความสัมพันธ์ของแอคเตอร์ทั้งหมดที่เกี่ยวข้องยกเว้นคำสั่งสร้างภาพมโนทัศน์ส่งให้ส่วนประมวลผลสร้างภาพมโนทัศน์ (Animation Engine) ตามลำดับที่ 1-3 ดังรูปที่ 4.31

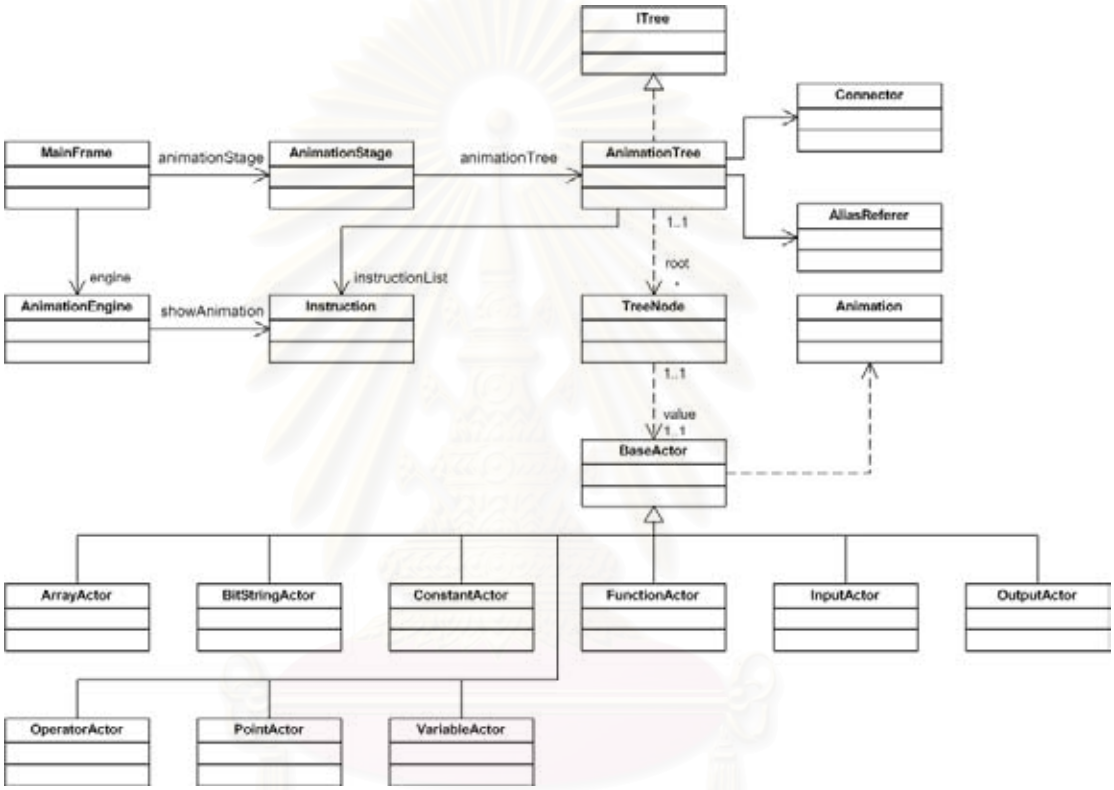
2) การแสดงภาพมโนทัศน์ เกิดขึ้นเมื่อผู้ใช้สั่งแสดงภาพมโนทัศน์การทำงานภายหลังจากการสร้างรูปโครงสร้างข้อมูลสำหรับการแสดงภาพมโนทัศน์นั้นเสร็จเรียบร้อยแล้ว



รูปที่ 4.31 แผนภาพกระแสข้อมูลการนำโครงสร้างข้อมูลและคำสั่งส่งให้กับส่วนประมวลผลสร้างภาพมโนทัศน์

ส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ทำการสร้างชุดคำสั่งสร้างภาพมโนทัศน์กำหนดไปพร้อมกับโครงสร้างข้อมูลให้ส่วนการประมวลผลสร้างภาพมโนทัศน์ทำงานตามคำสั่งตามลำดับที่ 4-6 ดังรูปที่ 4.31

แผนภาพคลาสไดอะแกรม และความสัมพันธ์ระหว่างคลาสต่างๆ เพื่อแสดงการออกแบบส่วนประมวลผลสร้างภาพมโนทัศน์ และความสัมพันธ์ระหว่างวัตถุ ดังรูปที่4.32



รูปที่ 4.32 แผนภาพคลาสไดอะแกรมแสดงความสัมพันธ์ระหว่างคลาสต่างๆ ของส่วนประมวลผลสร้างภาพมโนทัศน์

คลาสที่เกี่ยวข้องกับส่วนประมวลผลสร้างภาพมโนทัศน์ประกอบด้วยคลาสที่สำคัญดังต่อไปนี้

- คลาส Connector
- คลาส Animation
- คลาส AnimationEngine
- คลาส AnimationStage

- คลาส AliasReferer

คลาส Connector คือ คลาสสำหรับใช้สร้างเส้นเชื่อมความสัมพันธ์ของแอคเตอร์ และใช้แสดงค่าเคลื่อนไหวของข้อมูลในระหว่างแสดงภาพมโนทัศน์การทำงานของอัลกอริทึม

คลาส Animation คือ หน่วยของการกระทำที่แอคเตอร์แต่ละตัวจะกระทำเมื่อเกิดมีการเจาะจงให้คลาสแอคเตอร์ใดๆ แสดงจากคำสั่งที่ถูกป้อนให้ AnimationEngine คลาสนี้จะมีลักษณะเป็นการกระทำ แตกต่างกับคลาสอื่นๆ ในระบบ เนื่องจากผู้ออกแบบต้องการให้แอคเตอร์ทุกประเภทมีหน่วยการกระทำเป็นสิ่งเดียวกัน

เมทธอดที่สำคัญมีดังนี้

- init () ทำหน้าที่กำหนดค่าเริ่มต้นของอ็อบเจ็คแอคเตอร์

- animate (double p) ทำหน้าที่ให้แอคเตอร์กำหนดการกระทำของตนต่อหนึ่งช่วงเวลาที่ผ่านไป การกระทำในที่นี้ส่วนมากจะเป็นการเปลี่ยนตำแหน่งทีละพิกเซล โดยเมื่อเกิดการวาดรูปใหม่จะทำให้รูปที่ได้เปลี่ยนไปทีละน้อย เกิดเป็นภาพคลื่อนไหวในสายตาของผู้ใช้

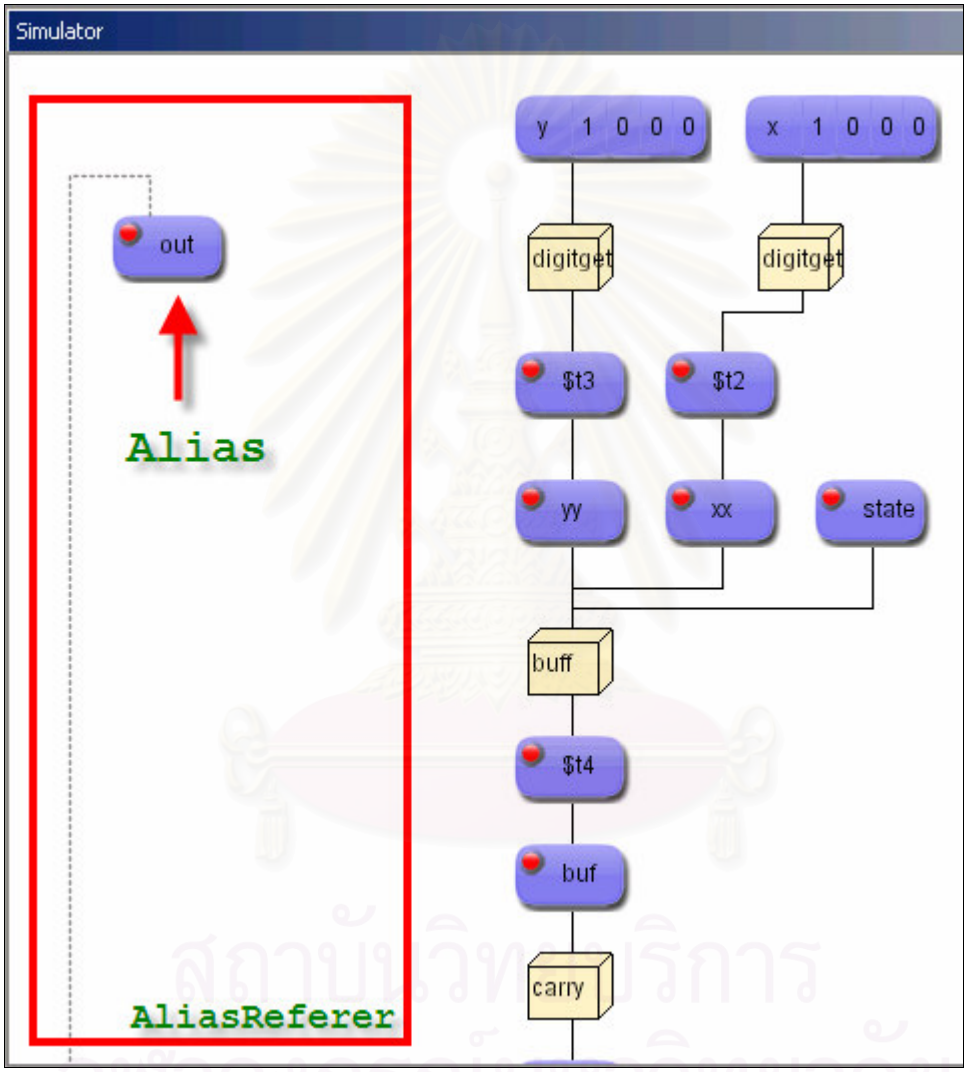
- finish () มีไว้เพื่อให้แอคเตอร์ทำการล้างค่าของตัวแปรที่จำเป็นเมื่อการแสดงสิ้นสุด

คลาส AnimationStage คือ คลาสสืบทอดคุณสมบัติมาจาก JComponent ซึ่งเป็นคลาสที่ใช้ในการสร้างส่วนแสดงภาพมโนทัศน์ โดยใช้คลังโปรแกรมของ Java Swing คลาสนี้มีหน้าที่แสดงผลที่เกิดจากการแสดงของแอคเตอร์ทั้งหมดซึ่งถูกเก็บการอ้างอิงค่าใน AnimationTree และถูกส่งผ่านมาที่ AnimationStage เมทธอดที่สำคัญมีดังนี้

- paintComponent (Graphics g) ทำหน้าที่วาดพื้นหลังซึ่งเป็นพื้นที่สีขาวและวาดแอคเตอร์ที่เก็บอยู่ภายใน AnimationTree

- paintActors (Graphics2D g2d) ทำหน้าที่วาดแอคเตอร์ทั้งหมด โดยที่แอคเตอร์แต่ละประเภทจะรู้ว่าตัวเองมีหน้าที่ต้องวาดออกมาเป็นลักษณะอย่างไร ดังนั้นเมทธอดนี้จึงส่งความรับผิดชอบไปให้แอคเตอร์แต่ละประเภทวาดกันเอง

คลาส AliasReferer คือ คลาสทำหน้าที่แสดงผลการอ้างอิงตัวแปรตัวเดียวกันภายในโปรแกรมต้นฉบับหรือเรียกว่า Alias โดยจะกำหนดไว้ทางด้านซ้ายของส่วนแสดงภาพมโนทัศน์เพื่อใช้เชื่อมความสัมพันธ์ในการอ้างอิงค่าตัวแปรให้ทำงานต่อเนื่องในขณะแสดงภาพมโนทัศน์ ดังรูปที่ 4.33



รูปที่ 4.33 การแสดงผลของการอ้างอิงตัวแปรแบบใช้ Alias

คลาส AnimationEngine คือ คลาสทำหน้าที่กำกับการแสดงของแอคเตอร์ การเคลื่อนไหวเกิดจากตำแหน่งของแอคเตอร์เปลี่ยนไปเนื่องจากการวาดรูปใหม่ ซึ่งสามารถเพิ่มหรือลดความเร็วได้ จากการที่เพิ่มหรือลดความเร็วในการวาดรูปใหม่ หากวาดรูปช้าเท่าไหร่ ในสายตา

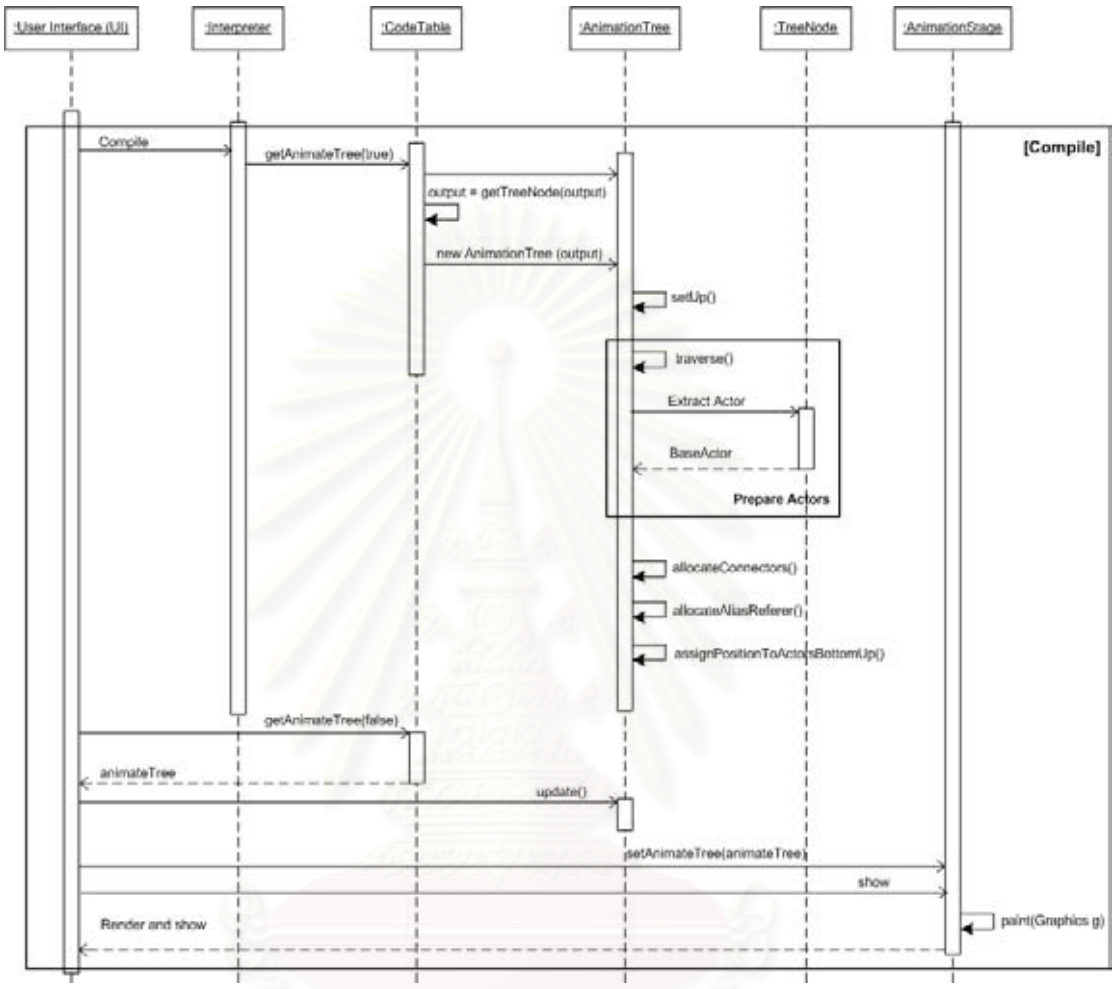
ของผู้ใช้จะรู้สึกเหมือนกับการเคลื่อนไหวนั้นดำเนินไปอย่างช้าๆ ในทางตรงกันข้าม หากเราวาดรูปใหม่อย่างรวดเร็วก็จะทำให้ผู้ใช้รู้สึกว่าการเคลื่อนไหวนั้นดำเนินไปอย่างรวดเร็วและต่อเนื่อง เมทธอดที่สำคัญมีดังนี้

- showAnimation (Instruction instr) ทำหน้าที่กำกับการแสดง (Animation) ของ BaseActor ซึ่งถูกเจาะจงมาผ่าน Instruction โดยผู้ออกแบบอาศัยหลักการมัลติเทรดดิ้ง (MultiThreading) ในการคำนวณหาเวลาที่เหมาะสม จากค่าของแถบเลื่อนปรับเปลี่ยนความเร็ว บนส่วนประสานงานผู้ใช้ ซึ่งผู้ใช้สามารถปรับได้อย่างอิสระจากนั้นนำค่าที่คำนวณได้มาสั่งให้ Thread sleep

- AnimationStage จะสัมพันธ์กับ AnimationTree โดยที่ AnimationTree จะเก็บการอ้างอิงข้อมูลของแอคเตอร์ทั้งหมด ส่วนการแสดงภาพมโนทัศน์จะถ่ายทอดคำสั่งให้แสดงผลไปยัง AnimationTree จากนั้นแอคเตอร์แต่ละตัวจะรู้ว่าตนเองจะต้องถูกแสดงผลออกมาเป็นลักษณะใด นอกจากนี้พื้นที่ส่วนแสดงภาพมโนทัศน์ยังทำหน้าที่ควบคุมจุดสนใจบนหน้าจอ (Focusing) เมื่อเกิดการแสดงภาพมโนทัศน์ แอคเตอร์ที่ขณะนั้นกำลังจะเริ่มการแสดงแต่อยู่พ้นสายตาของผู้ใช้ส่วนการแสดงภาพมโนทัศน์จะเลื่อนไปที่ตำแหน่งที่แสดงอยู่โดยอัตโนมัติ

- แนวคิดของการควบคุมจุดสนใจบนหน้าจอมาจากการมองหน้าจอทั้งหมด เป็นหลายหน้าจอสมมุติ (virtual screen) และจำพิกัดเริ่มต้นของแต่ละหน้าจอสมมุติ เมื่อแอคเตอร์ที่กำลังทำการแสดงอยู่นอกเหนือพิกัดที่กำหนดไว้ในหน้าจอสมมุติปัจจุบัน ส่วนแสดงภาพมโนทัศน์จะคำนวณว่าหน้าจอสมมุติที่กำลังจะเปลี่ยนไปนั้นสัมพันธ์กับหน้าจอสมมุติปัจจุบันอย่างไรบ้างจากทางด้านบน ,ด้านล่าง,ด้านซ้าย ,ด้านขวา

แผนภาพซีเควนซ์แสดงการทำงานส่วนประมวลผลสร้างภาพมโนทัศน์ มีลำดับของกิจกรรม ดังนี้

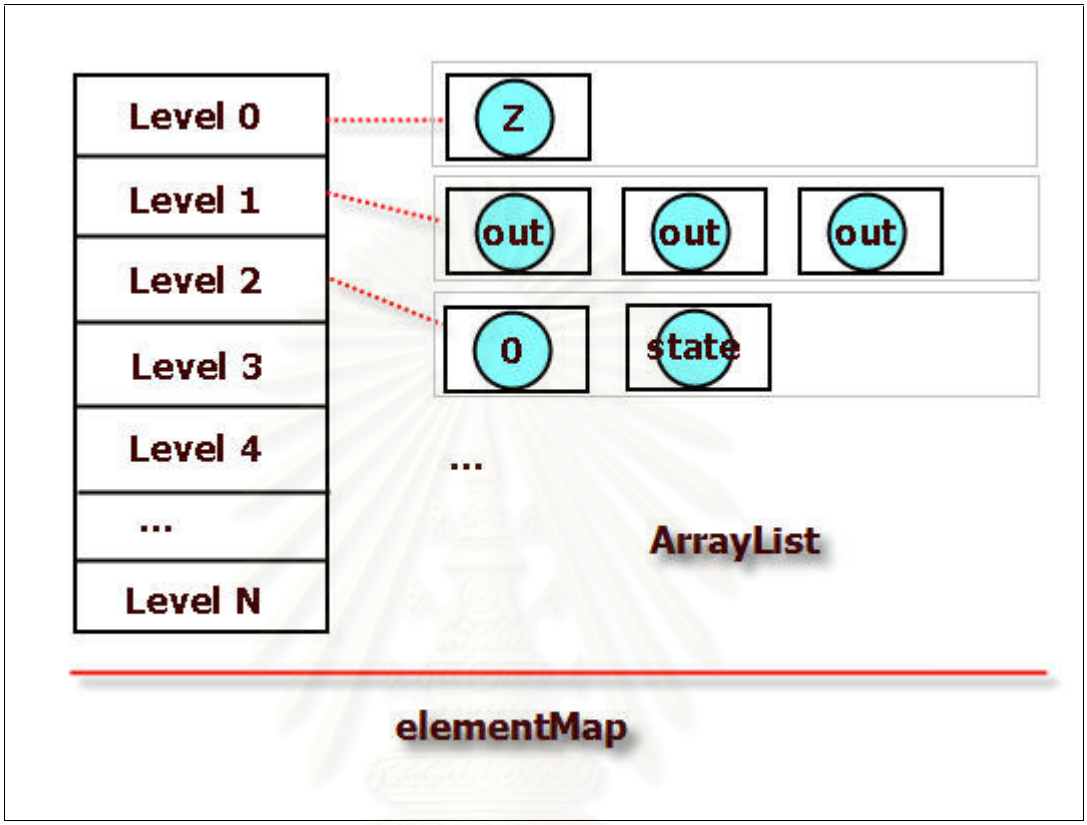


รูปที่ 4.34 แผนภาพซีเควนซ์แสดงการทำงานของส่วนประมวลผลสร้างภาพมโนทัศน์

1. แสดงคำสั่งอินเตอร์มิเดียทในรูปแบบกราฟิก มีขั้นตอนการทำงานดังนี้

1.1 AnimationTree ถูกสร้างจากส่วนอินเตอร์มิเดียทอินเทอร์พรีเตอร์ในขั้นตอนการคอมไพล์โปรแกรมต้นฉบับ โดยใช้ TreeNode ตัวแปรผลลัพธ์เป็นพารามิเตอร์ จากนั้นเรียกเมทธอด traverse () เพื่อท่องในแต่ละ TreeNode จากการอ้างอิงค่าที่เก็บอยู่ใน AnimationTree จนครบทุก TreeNode โดยที่ภายใน TreeNode จะบรรจุข้อมูลของแอคเตอร์ไว้ ในแต่ละครั้งที่มีการท่องใน TreeNode จะมีการนำ TreeNode ที่บรรจุแอคเตอร์เก็บไว้ในตารางแฮช elementMap ดังรูปที่ 4.35 โดยที่มีคำหลักเป็นระดับชั้นของ TreeNode ปัจจุบัน และค่าเป็น

ArrayList ที่เก็บแอคเตอร์ทั้งหมดในระดับชั้นปัจจุบัน ในการคำนวณระดับชั้นปัจจุบันจะคิดจากความห่างของ TreeNode ปัจจุบันถึงส่วนของ TreeNode ราก



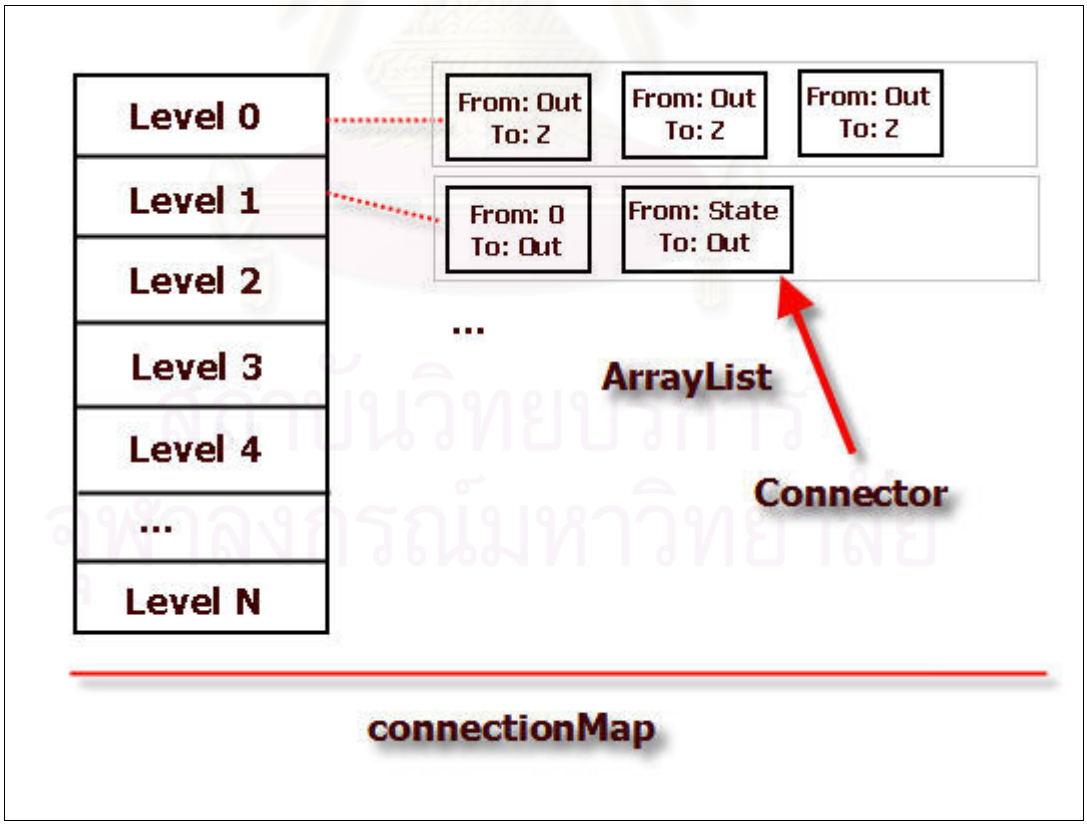
รูปที่ 4.35 การจัดเก็บข้อมูลใน elementMap

1.2 เมื่อ elementMap มีข้อมูลลำดับชั้น รวมถึง Treenode ที่ใช้เก็บแอคเตอร์แต่ละตัวทั้งหมดแล้ว จากนั้นจะเรียกเมทธอด allocateConnectors () เพื่อสร้างแอคเตอร์ Connector โดยที่แอคเตอร์ Connector เปรียบได้กับตัวกระทำการเครื่องหมาย “=” เพื่อเป็นการส่งค่าข้อมูลด้านขวาไปยังด้านซ้าย การสร้างแอคเตอร์ Connector นั้นทำได้โดยตรวจสอบความสัมพันธ์ระหว่างแอคเตอร์ที่อยู่ต่างระดับกัน ข้อมูลของ Connector ทั้งหมดจะถูกเก็บไว้ในตารางแฮช เพื่อนำไปใช้ในการวาดและแสดงภาพมโนทัศน์ต่อไป

อัลกอริทึมการสร้าง Connector หรือเส้นเชื่อม มีขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1 วนซ้ำใน elementMap แต่ละระดับชั้น โดยค้นหาจากชั้นบนสุด คือ ชั้นของ TreeNode ราก หรือชั้นของตัวแปรผลลัพธ์ในมุมมองผู้ใช้ จนถึงชั้นล่างสุด หรือชั้นของตัวแปรนำเข้าในมุมมองผู้ใช้ โดยใช้คำหลักเป็นระดับชั้นปัจจุบัน เพื่อเรียกใช้ค่า Treenode ที่เก็บอยู่ใน ArrayList ซึ่งภายในบรรจุแอคเตอร์ในระดับชั้นปัจจุบันนำออกมาใช้งาน

ขั้นตอนที่ 2 วนซ้ำใน ArrayList ที่เก็บแอคเตอร์ทั้งหมดในระดับชั้นปัจจุบัน เพื่อตรวจสอบว่าแอคเตอร์ตัวปัจจุบันมี TreeNode ลูกตัวใดบ้าง โดยที่ความสัมพันธ์ระหว่าง TreeNode แม่ กับ TreeNode ลูก จะนำมาใช้สร้างแอคเตอร์ Connector โดยที่ TreeNode แม่จะเป็นปลายด้านหนึ่งของแอคเตอร์ Connector และปลายอีกด้านหนึ่งคือ TreeNode ลูก จากนั้นเก็บแอคเตอร์ Connector ที่สร้างขึ้นใน connectionMap โดยใช้คำหลักเป็นค่าระดับชั้นของ TreeNode แม่ เนื่องจากจำเป็นต้องมีการค้นหาแอคเตอร์ Connector ที่จะทำการแสดงอีกครั้งในการแสดงภาพมโนทัศน์ ดังรูปที่ 4.36



รูปที่ 4.36 การจัดเก็บข้อมูลใน connectionMap

1.3 เมทธอด allocateAliasReferer () จะถูกเรียกเพื่อทำการหาตัวแปรที่มีชื่อซ้ำและอยู่ต่างลำดับชั้นหรือเรียกอีกอย่างว่า Alias การหา Alias ทำได้โดยการเปรียบเทียบแอคเตอร์ที่อยู่ต่างระดับชั้นกัน

อัลกอริทึมการหา Alias มีขั้นตอนดังต่อไปนี้

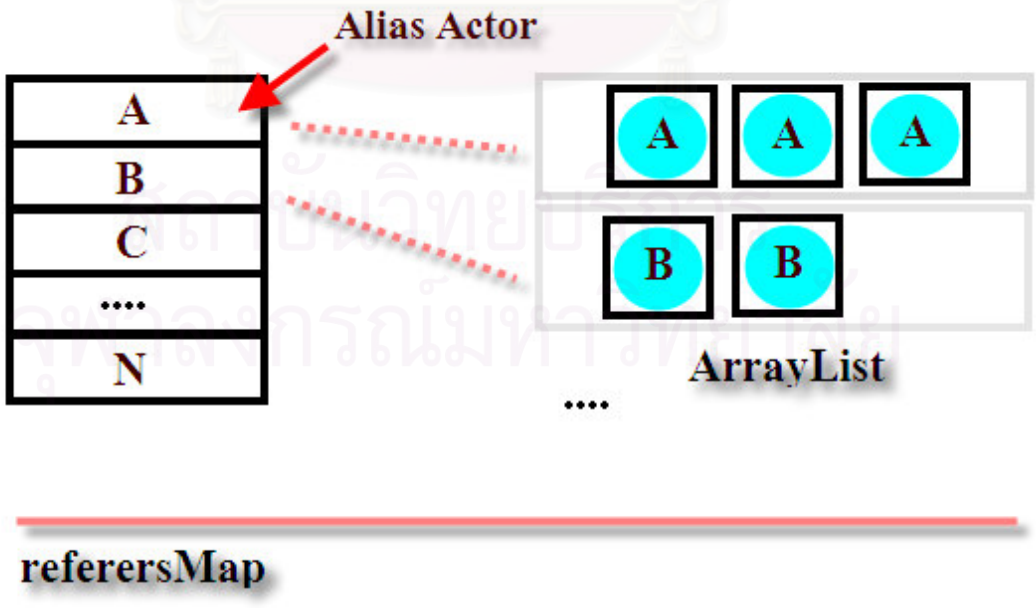
ขั้นตอนที่ 1 วนซ้ำใน elementMap แต่ละระดับชั้น โดยไล่จากชั้นบนสุดจนถึงชั้นล่างสุดใช้คำหลักเป็นระดับชั้นปัจจุบันเพื่อดึงค่า ArrayList ที่เก็บแอคเตอร์ทั้งหมดในระดับชั้นปัจจุบันออกมาใช้

ขั้นตอนที่ 2 วนซ้ำใน ArrayList ที่เก็บแอคเตอร์ทั้งหมดในระดับชั้นปัจจุบัน จากนั้นเทียบกับแอคเตอร์ทั้งหมดในชั้นถัดไป หากชื่อของแอคเตอร์ซ้ำกันหมายความว่าเกิด Alias ขึ้นมาแล้ว

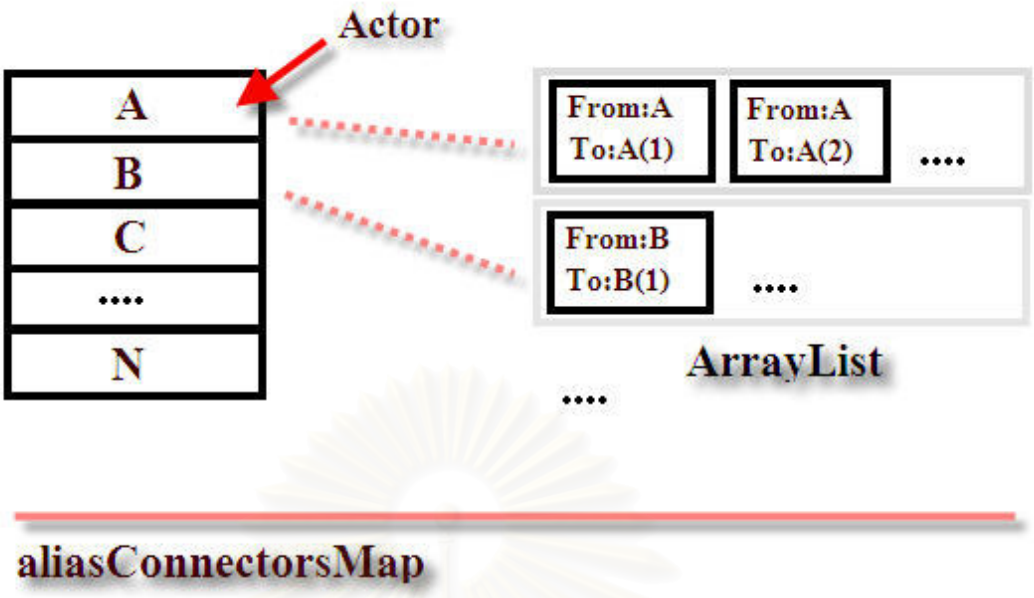
1.4 หลังจากหา Alias ได้ครบแล้วจะถูกกำหนดให้กับ AliasReferer จัดเก็บไว้ในตารางแฮช referersMap ดังรูปที่ 4.37 เพื่อนำไปใช้วาดไว้ทางด้านซ้ายของ AnimationStage

อัลกอริทึมการแสดงผลของ AliasReferer มีขั้นตอนดังต่อไปนี้

ขั้นตอนที่ 1 ในเมทธอด AliasReferer.setup () จะมีการสร้างแอคเตอร์ชนิดเดียวกับที่พบใน ArrayList ที่หามาได้ก่อนหน้านี้ โดยสร้างหนึ่งแอคเตอร์ต่อหลายตัวที่เกิดการซ้ำกัน



รูปที่ 4.37 การจัดเก็บข้อมูลใน referersMap

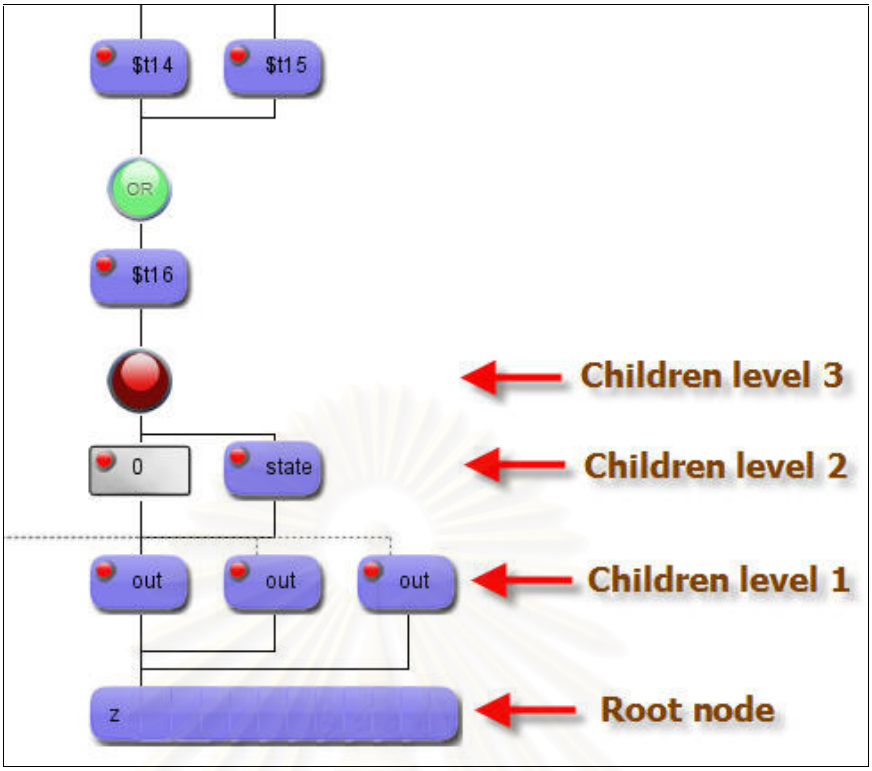


รูปที่ 4.38 การจัดเก็บข้อมูลใน aliasConnectorsMap

ขั้นตอนที่ 2 กำหนดตำแหน่งให้กับแอคเตอร์ที่ถูกสร้างขึ้นจากขั้นตอนที่แล้ว โดยเพิ่มค่า y แบบคงที่เพื่อให้แอคเตอร์เรียงจากตัวบนสุดไปจนถึงล่างสุด

ขั้นตอนที่ 3 สร้าง Connector จากแอคเตอร์ที่ถูกสร้างขึ้นใหม่ไปยังแอคเตอร์ที่เป็น Alias (ตัวแปรที่มีชื่อซ้ำกัน) ใน elementMap จัดเก็บไว้ในตารางแฮช aliasConnectorsMap ดังรูปที่ 4.38 เพื่อใช้เป็นตัวเชื่อมในการแสดงภาพมโนทัศน์ความสัมพันธ์

1.5 เมทธอด assignPositionToActorsBottomUp () จะถูกเรียกเพื่อทำการกำหนดตำแหน่งของแอคเตอร์แต่ละตัวโดยคำนวณตำแหน่งจากล่างขึ้นบน จากตัวแปรผลลัพธ์มายังตัวแปรนำเข้า เนื่องจากส่วนล่างสุดในมุมมองของผู้ใช้คือ TreeNode ราก ตำแหน่งของแอคเตอร์แต่ละตัว คำนวณจากระดับชั้นและลำดับของแอคเตอร์ในชั้นปัจจุบัน โดยที่ในแต่ละชั้นพิกัดแกน y จะมีค่าเท่ากันเสมอ แอคเตอร์ที่มีลำดับน้อยกว่าจะมีค่าแกน x ที่น้อยกว่า แล้วไล่ไปจนสุดในแต่ละชั้น เมื่อมีการเลื่อนระดับไปชั้นถัดไปจะเพิ่มพิกัดแกน y และล้างค่าพิกัดแกน x ไปที่ตำแหน่งแรก ทำอย่างนี้ไปจนครบแอคเตอร์ทุกลำดับ และทุกระดับชั้น ดังรูปที่ 4.39



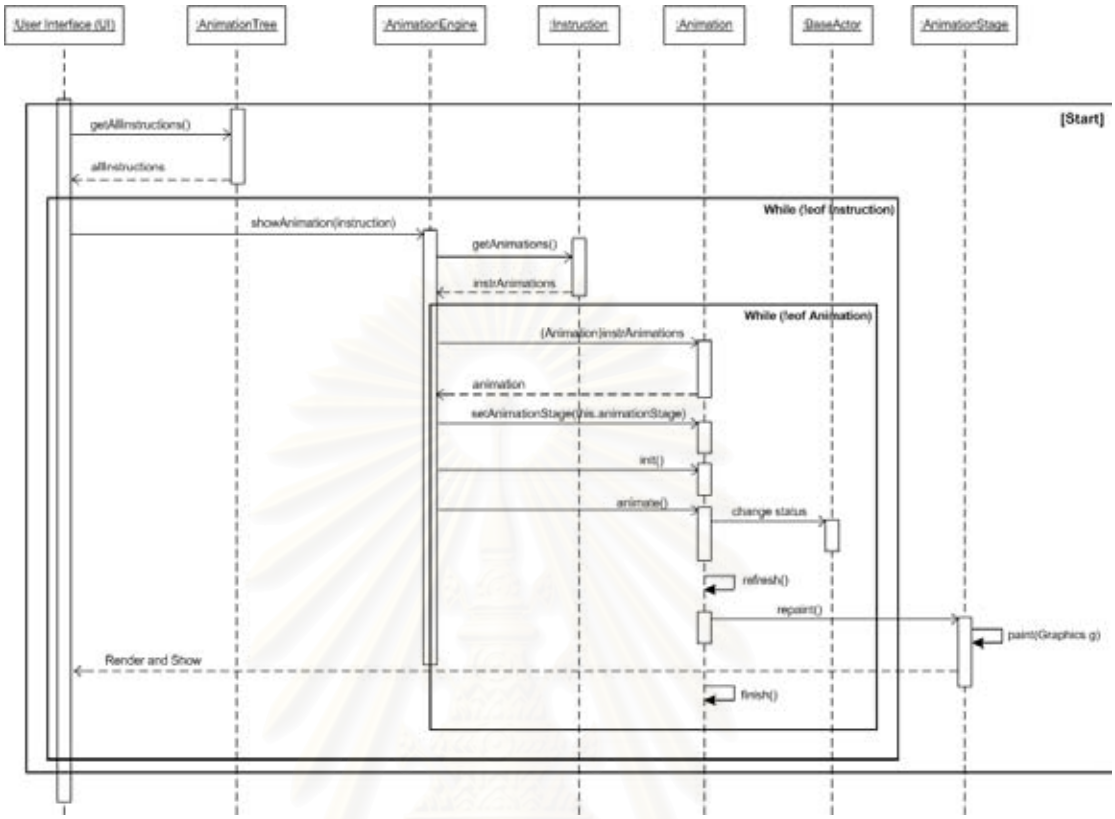
รูปที่ 4.39 การกำหนดตำแหน่งแอคเตอร์

1.6 หลังจากที่ได้ตำแหน่งของแอคเตอร์แต่ละตัวแล้ว อินเตอร์มิเดียทอินเทอร์พรีเตอร์จะเรียกเมทธอด update () ใน AnimationTree เพื่อจัดตำแหน่งให้แอคเตอร์ PointActor เนื่องจากแอตเตอร์ PointActor ถูกสร้างจากอินเตอร์มิเดียทอินเทอร์พรีเตอร์ โดยไม่ได้คำนวณจากไวยากรณ์ของภาษา หากแต่เป็นจุดเชื่อมต่อการไหลของข้อมูล จึงถูกนำมาจัดตำแหน่งหลังสุด

1.7 ข้อมูลในโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์ที่ประมวลผลแล้วจะถูกกำหนดให้ AnimationStage ผ่านเมทธอด setAnimaitionTree ()

1.8 เมทธอด paintComponent () ในคลาส AnimationStage เป็นเมทธอดที่สืบทอดมาจากคลาส JComponent ซึ่งจะถูกเรียกโดยอัติโนมัติทุกครั้งที่มีการเปลี่ยนแปลงค่าในส่วนแสดงภาพมโนทัศน์ เมื่อเมทธอดนี้ถูกเรียกใช้ AnimationStage จะทำการสั่งคำสั่งไปที่แอคเตอร์แต่ละตัวให้เรียกเมทธอด paintActor() โดยแอคเตอร์แต่ละตัวจะรู้ว่าต้องทำการวาดตัวเองลงบนผืนผ้ากราฟิกได้อย่างไร โดยอาศัยหลักการ polymorphism

2. สร้างภาพมโนทัศน์ เป็นขั้นตอนเกิดขึ้นหลังจากอินเตอร์มิเดียทอินเทอร์พรีเตอร์สร้างคำสั่งการแสดงภาพมโนทัศน์ของแอคเตอร์เสร็จเรียบร้อยแล้ว มีขั้นตอนการทำงานดังนี้



รูปที่ 4.40 แผนภาพซีเควนซ์แสดงการทำงานของส่วนประมวลผลสร้างภาพมโนทัศน์

2.1 เรียกใช้คำสั่งการแสดงภาพมโนทัศน์ของแอคเตอร์ทั้งหมดจากโครงสร้างข้อมูลสำหรับสร้างภาพมโนทัศน์ ผ่านเมทธอด getAllInstruction () โดยขั้นตอนนี้จะถูกกระทำในเทรดย่อยแบ่งแยกการทำงานออกจากส่วนประสานงานผู้ใช้โดยจะต้องทำงานไปพร้อมๆ กับส่วนส่วนติดต่อผู้ใช้ หากไม่มีการแยกเทรดออกจากกัน จะทำให้ส่วนติดต่อกับผู้ใช้ไม่สามารถทำงานด้วยการขัดจังหวะจากคำสั่งอื่นๆ ได้

2.2 จากนั้นเรียกใช้คำสั่งสำหรับสร้างภาพมโนทัศน์จาก Instruction ผ่านเมทธอด showAnimation () ซึ่งใช้กำกับการแสดงของแอคเตอร์ โดยที่แอคเตอร์แต่ละตัวเมื่อถูกกำหนดลำดับให้แสดงแล้วจะมีการเรียกเมทธอด ผ่าน animate () จาก BaseActor เพื่อดึงขั้นตอนการแสดงออกมาจากแอตเตอร์ตัวที่ถูกกำหนด

ในการแสดงของแอคเตอร์แต่ละตัวจะสัมพันธ์กับเมทธอด paintActor () เพื่อให้แอคเตอร์ทำการวาดตัวเองลงใน AnimationStage แต่มีการเปลี่ยนแปลงทีละน้อย โดยอาศัยหลักการเดียวกับการวาดรูปซ้ำๆ กันในกระดาษหลายๆ ใบ โดยแต่ละใบมีการขยับตำแหน่งของรูปทีละน้อย เมื่อมีการพลิกกระดาษด้วยความเร็วระดับหนึ่งจะเห็นเสมือนภาพวาดมีการเคลื่อนไหว

บทที่ 5

การพัฒนาและทดสอบเครื่องมือ

ในบทนี้กล่าวถึงสภาพแวดล้อม และโครงสร้างของเครื่องมือที่ใช้ในการพัฒนาเครื่องมือสำหรับแสดงภาพมโนทัศน์การคำนวณแบบเชื่อมตรงระดับดิจิต วิทยานิพนธ์นี้ได้พัฒนาโปรแกรมที่ชื่อว่า OAS Simulation ด้วยภาษา Java ในรูปแบบของวินโดวส์แอพพลิกเคชั่น และใช้เครื่องมือการสร้างตัวแปลภาษาโดยใช้โครงสร้างเชิงวัตถุพัฒนาด้วย SableCC และสร้างตัวแปลชุดคำสั่งอินเตอร์มีเดียทด้วย Grammatiga เครื่องมือใช้อุปกรณ์ในการพัฒนาคือ Eclipse 3.0 โดยหน้าจอ และวิธีการใช้งานจะอธิบายอยู่ในภาคผนวก ง. รายละเอียดการพัฒนาและทดสอบเครื่องมือ มีดังนี้

5.1 สภาพแวดล้อมที่ใช้ในการพัฒนาเครื่องมือ

1) ฮาร์ดแวร์ (Hardware)

1.1) เครื่องคอมพิวเตอร์แบบพีซี (PC) หน่วยประมวลผลอินเทลเพนเทียมเอ็ม 1.5 กิกะเฮิร์ทซ์ (Intel Pentium M 1.5 GHz)

1.2) หน่วยความจำสำรอง (RAM)512 เมกะไบต์ (512 MB)

1.3) ฮาร์ดดิสก์ (Hard disk) 4 0กิกะไบต์ (40 GB)

2) ซอฟต์แวร์ (Software)

2.1) ระบบปฏิบัติการ (Operating system) ไมโครซอฟท์วินโดวส์เอ็กซ์พี โพรเฟซชันแนล เซอร์วิสแพค 2 (Microsoft Windows XP Professional Service Pack 2)

2.2) เครื่องมือที่ใช้พัฒนา

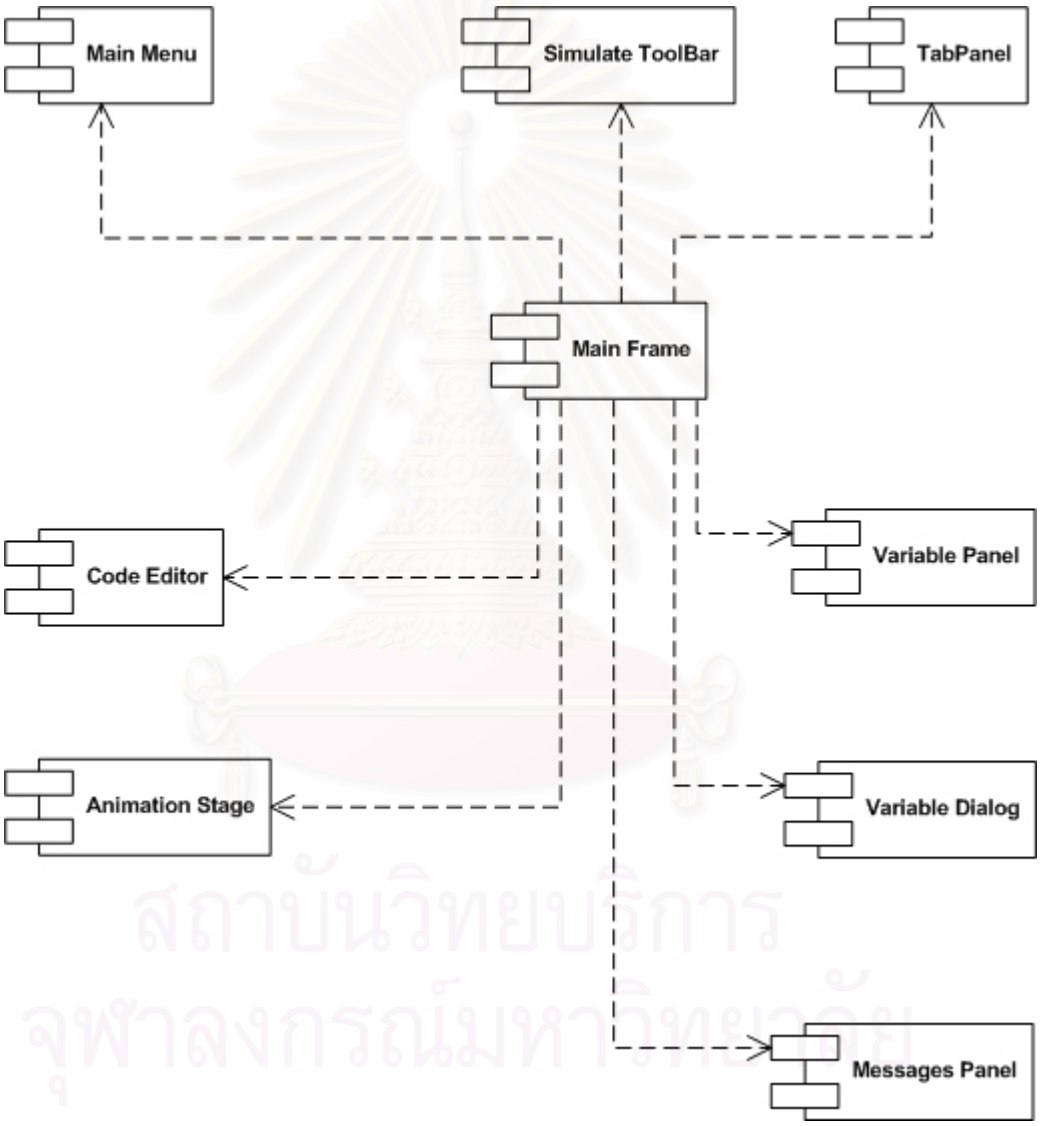
- เอคลิปซ์ (Eclipse 3.0)
- SABLECC เครื่องมือสร้างโครงสร้างตัวแปลภาษาแบบเชิงวัตถุ (Sablecc : object-oriented framework for generating compilers)
- Grammatiga เครื่องมือสร้างตัวแปลคำสั่งทีละคำสั่ง (Grammatica : Intepreter Tool)

2.3) ภาษาที่ใช้พัฒนา

- Java (Java Language)

5.2 โครงสร้างของเครื่องมือ

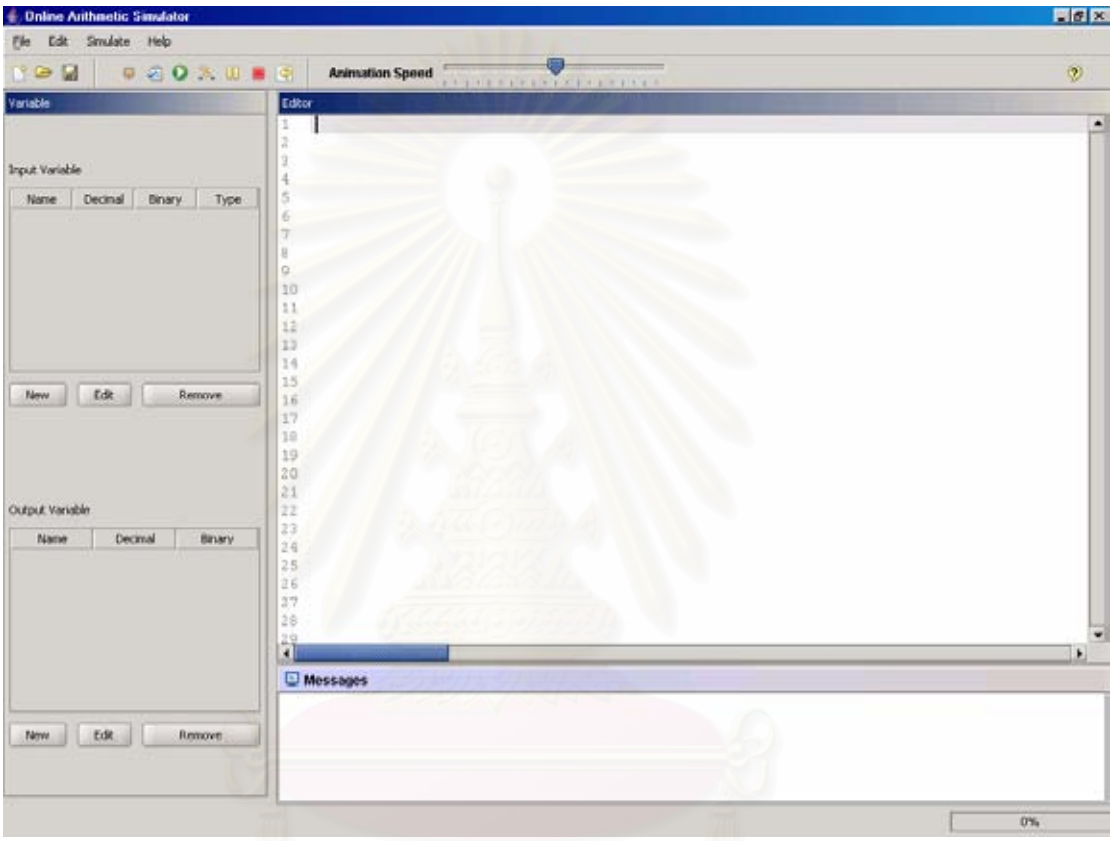
โครงสร้างของเครื่องมือจะอธิบายองค์ประกอบ และหน้าที่การทำงานของเครื่องมือ เช่น เมนูหลัก (Main Menu), ทูลบาร์ควบคุมภาพมโนทัศน์ (Simulate Toolbar) เป็นต้น โดยจะแสดงด้วยแผนภาพแสดงส่วนประกอบ (Component diagram) ซึ่งเป็นแผนภาพแสดงความสัมพันธ์ของส่วนประกอบของเครื่องมือ ดังรูปที่ 5.1



รูปที่ 5.1 แผนภาพความสัมพันธ์ส่วนประกอบของเครื่องมือสร้างภาพมโนทัศน์

รายละเอียดโครงสร้างของเครื่องมือ เป็นดังนี้

1. ส่วนหน้าจอหลัก (Main Frame) คือหน้าจอหลักของเครื่องมือสร้างภาพมโนทัศน์สำหรับผู้ออกแบบอัลกอริทึม แสดงดังรูปที่ 5.2 ส่วนหน้าจอหลักถูกออกแบบด้วย Façade Pattern [18] [19] เพื่อเป็นส่วนสำหรับเชื่อมส่วนประกอบเครื่องมือย่อยอื่นๆ และถ่ายทอดคำสั่งไปยังส่วนประกอบที่เกี่ยวข้องเมื่อมีการสั่งงานจากผู้ใช้

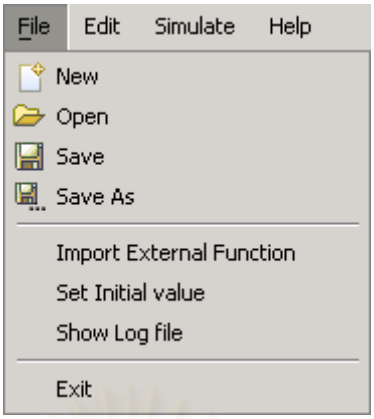


รูปที่ 5.2 หน้าจอหลักของเครื่องมือ

2. เมนูหลัก (Main Menu) คือส่วนที่ใช้ควบคุมการทำงานของเครื่องมือ เมนูหลักมีทั้งหมด 4 เมนูย่อย ดังนี้

2.1) เมนูไฟล์ (File Menu) เป็นเมนูที่ใช้ควบคุมการทำงานของแฟ้มข้อมูลโปรแกรมต้นฉบับ ประกอบด้วยส่วนประกอบ ดังรูปที่ 5.3

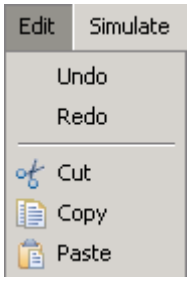
- สร้างโปรแกรมต้นฉบับใหม่ (New) คือคำสั่งให้สร้างแฟ้มข้อมูลโปรแกรมต้นฉบับใหม่



รูปที่ 5.3 เมนูไฟล์

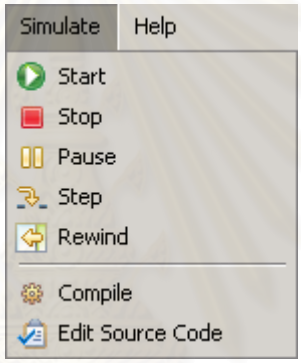
- เปิดแฟ้มข้อมูล (Open) คือ เมนูเปิดแฟ้มข้อมูลโปรแกรมต้นฉบับที่มีอยู่แล้ว หากมีการเปิดแฟ้มข้อมูลโปรแกรมต้นฉบับด้วยคำสั่งนี้เครื่องมือจะจดจำตำแหน่งของที่เก็บข้อมูลครั้งล่าสุดไว้
- บันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับ (Save) คือเมนูบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับในขณะที่ผู้ใช้กำลังทำการแก้ไขบนพื้นที่ส่วนการเขียนโปรแกรมต้นฉบับ
- บันทึกแฟ้มข้อมูลเป็นชื่อแฟ้มข้อมูลอื่น (Save as) คือ เมนูบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับที่ผู้ใช้กำลังทำการแก้ไขบนพื้นที่ส่วนการเขียนโปรแกรมต้นฉบับเป็นแฟ้มข้อมูลอื่น
- การนำเข้าฟังก์ชันภายนอก (Import External Function) คือ เมนูนำเข้าฟังก์ชันภายนอกที่ผู้ออกแบบอัลกอริทึมแยกการจัดเก็บจากฟังก์ชันหลัก
- กำหนดค่าตั้งต้นของตัวแปรนำเข้า (Set Initial value) คือ เมนูกำหนดค่าเริ่มต้นของตัวแปรนำเข้า
- แสดงค่าการทำงานของตัวแปรในแต่ละขั้นตอน (Show Log file) คือ เมนูแสดงค่าการทำงานของตัวแปร
- แฟ้มข้อมูลล่าสุดที่ใช้ (Recently Opened Files) รายชื่อตำแหน่งและชื่อของแฟ้มข้อมูลโปรแกรมต้นฉบับที่ผู้ใช้ทำการแก้ไขล่าสุด โดยเรียงลำดับตามเวลาการแก้ไข
- ออกจากระบบ (Exit) คือคำสั่งให้เครื่องมือหยุดการทำงาน

2.2) เมนูแก้ไข (Edit Menu) คือเมนูเกี่ยวกับคำสั่งแก้ไขแฟ้มข้อมูลโปรแกรมต้นฉบับด้วยคำสั่ง undo, redo, cut, copy, paste ดังรูปที่ 5.4



รูปที่ 5.4 เมนูแก้ไข

2.3) เมนูควบคุมการแสดงภาพมโนทัศน์ (Simulate) คือ เมนูควบคุมการแสดงภาพมโนทัศน์การทำงาน ประกอบด้วยเมนูย่อยดังรูป 5.5

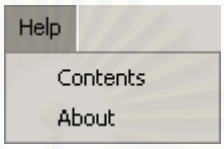


รูปที่ 5.5 เมนูควบคุมการแสดงภาพมโนทัศน์

- แปลโปรแกรม (Compile) คือเมนูสั่งตรวจสอบความถูกต้องของโปรแกรมต้นฉบับที่ผู้ออกแบบอัลกอริทึมป้อนในพื้นที่ส่วนการเขียนโปรแกรม
- แสดงภาพมโนทัศน์ (Start) คือเมนูสั่งแสดงภาพมโนทัศน์การทำงานของอัลกอริทึม
- หยุดแสดงภาพมโนทัศน์ (Stop) คือเมนูสั่งหยุดการแสดงภาพมโนทัศน์
- หยุดแสดงภาพมโนทัศน์ชั่วคราว (Pause) คือเมนูสั่งหยุดการแสดงภาพมโนทัศน์ชั่วคราว
- แสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่ง (Step) คือเมนูสั่งแสดงภาพมโนทัศน์โดยเริ่มจากโค้ดในบรรทัดตำแหน่งขณะที่กำลังแสดงภาพมโนทัศน์จนจบการทำงานของคำสั่งนั้น

- แสดงภาพมโนทัศน์การทำงานใหม่ (Rewind) คือเมนูสั่งหยุดแสดงภาพมโนทัศน์และแสดงภาพมโนทัศน์การทำงานใหม่
- แก้ไขโปรแกรมต้นฉบับ (Edit Source Code) คือ เมนูแก้ไขหรือเปลี่ยนแปลงโปรแกรมต้นฉบับ

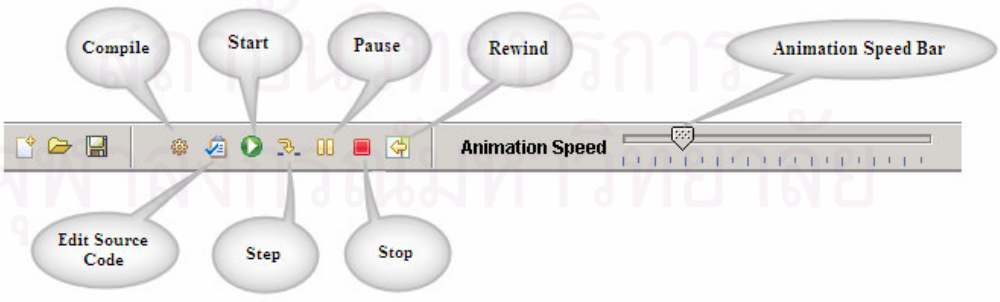
2.4 เมนูความช่วยเหลือ (Help) คือเมนูเกี่ยวกับการใช้โปรแกรม ประกอบด้วยเมนูย่อยดังรูป 5.6



รูปที่ 5.6 เมนูความช่วยเหลือ

- รายละเอียดการใช้ระบบ (Contents) คือเมนูแสดงการใช้งานโปรแกรม
- เกี่ยวกับระบบ (About) คือเมนูเกี่ยวกับเครื่องมือ

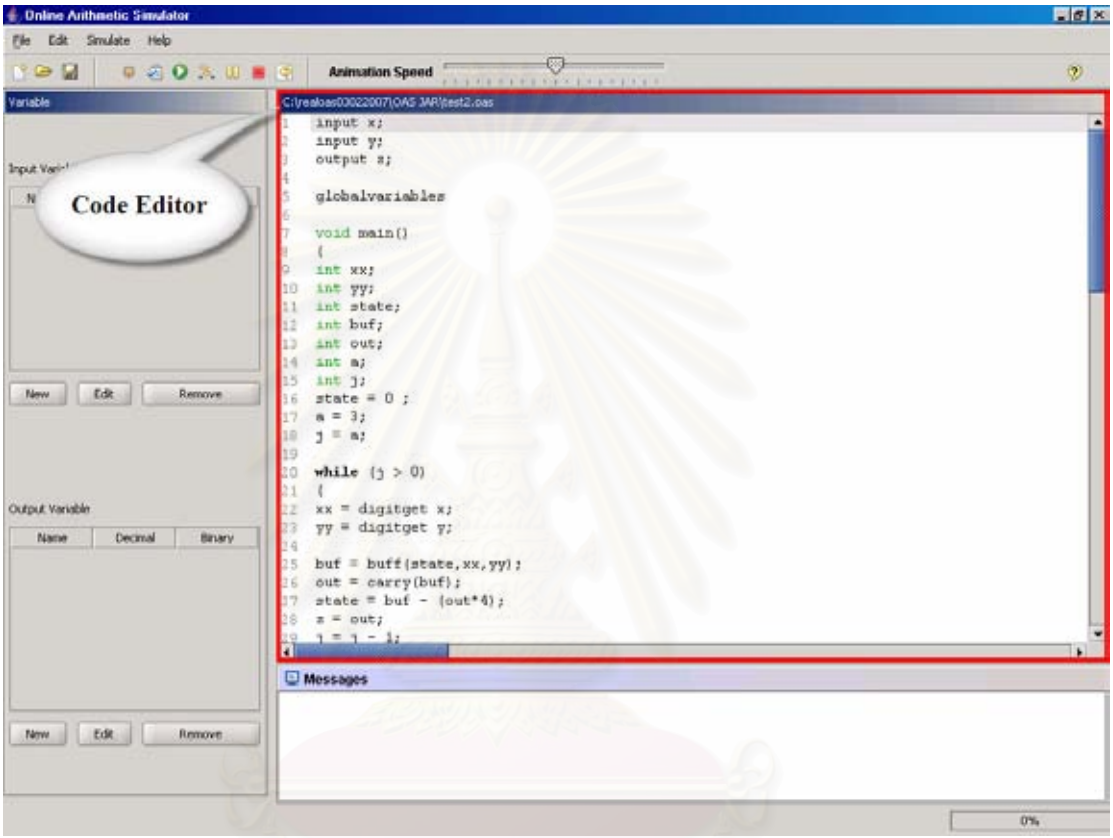
3. ทูลบาร์ (Toolbar) คือ ส่วนควบคุมเมนูไฟล์, เมนูควบคุมการแสดงภาพมโนทัศน์และส่วนควบคุมความเร็วในการแสดงภาพมโนทัศน์ (Simulation Speed Bar) การควบคุมความเร็วในการแสดงภาพมโนทัศน์ สามารถทำด้วยการเลื่อนแถบปรับเปลี่ยนความเร็ว โดยเลื่อนไปทางซ้ายจะเป็นการลดความเร็ว ในทางตรงกันข้ามถ้าเลื่อนไปทางขวาจะเป็นการเพิ่มความเร็วการแสดงภาพมโนทัศน์ ดังรูป 5.7



รูปที่ 5.7 ทูลบาร์ควบคุมการแสดงภาพมโนทัศน์

4. ส่วนการเขียนโปรแกรมต้นฉบับ (Code Editor) คือ ส่วนสำหรับเขียนหรือแก้ไขโปรแกรมต้นฉบับ ซึ่งเป็นส่วนที่ใช้จำไวยากรณ์คำสั่ง ช่วยให้ผู้ใช้มองเห็นไวยากรณ์ภาษาที่ระบุได้

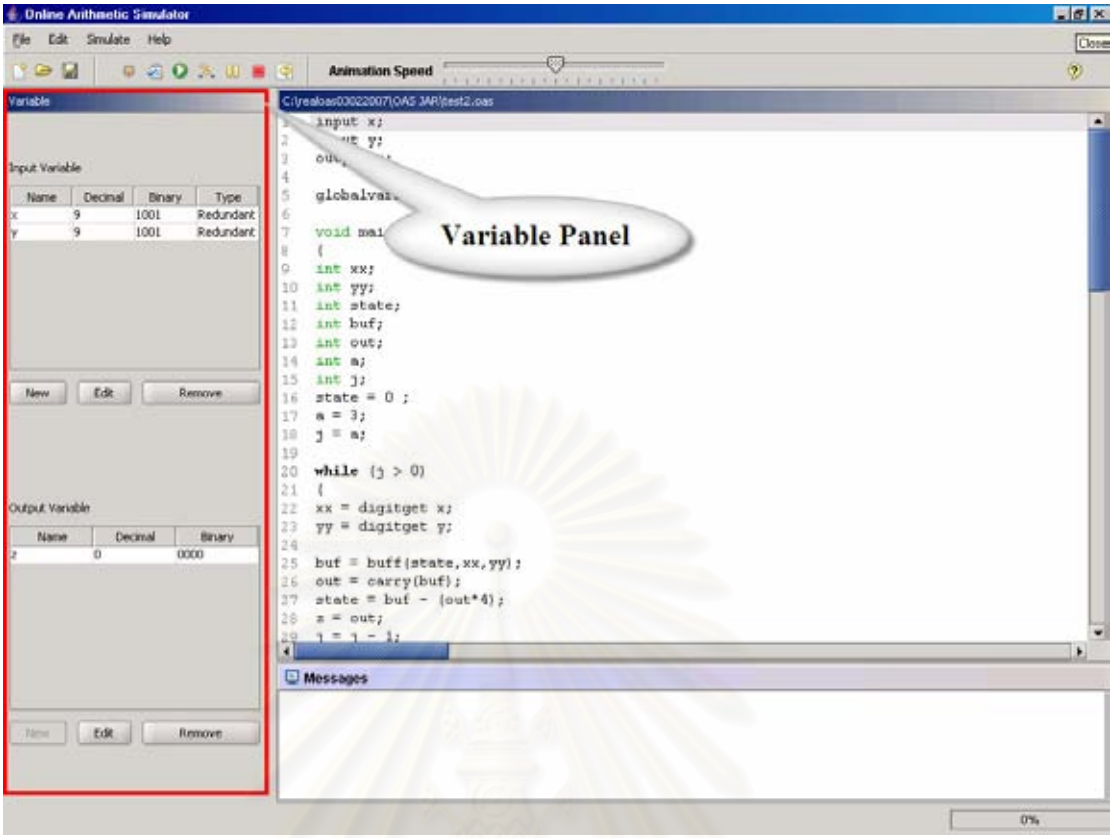
ชัดเจนขึ้น (Syntax highlighting) นอกจากนี้ส่วนการเขียนโปรแกรมต้นฉบับยังทำหน้าที่รับคำสั่งที่เกี่ยวข้องกับแฟ้มข้อมูลโปรแกรมต้นฉบับที่ผู้ใช้เขียนขึ้นอันได้แก่ การเปิดโปรแกรมต้นฉบับแฟ้มข้อมูล, การสร้างแฟ้มไฟล์ใหม่, การจัดเก็บแฟ้มข้อมูล, การจัดเก็บแฟ้มข้อมูลเป็นแฟ้มข้อมูลชื่ออื่น รวมทั้งการแถบสีบรรทัดที่เกี่ยวข้องกับการแสดงภาพมโนทัศน์ในขณะที่ผู้ใช้สั่งแสดงผล ดังรูปที่ 5.8



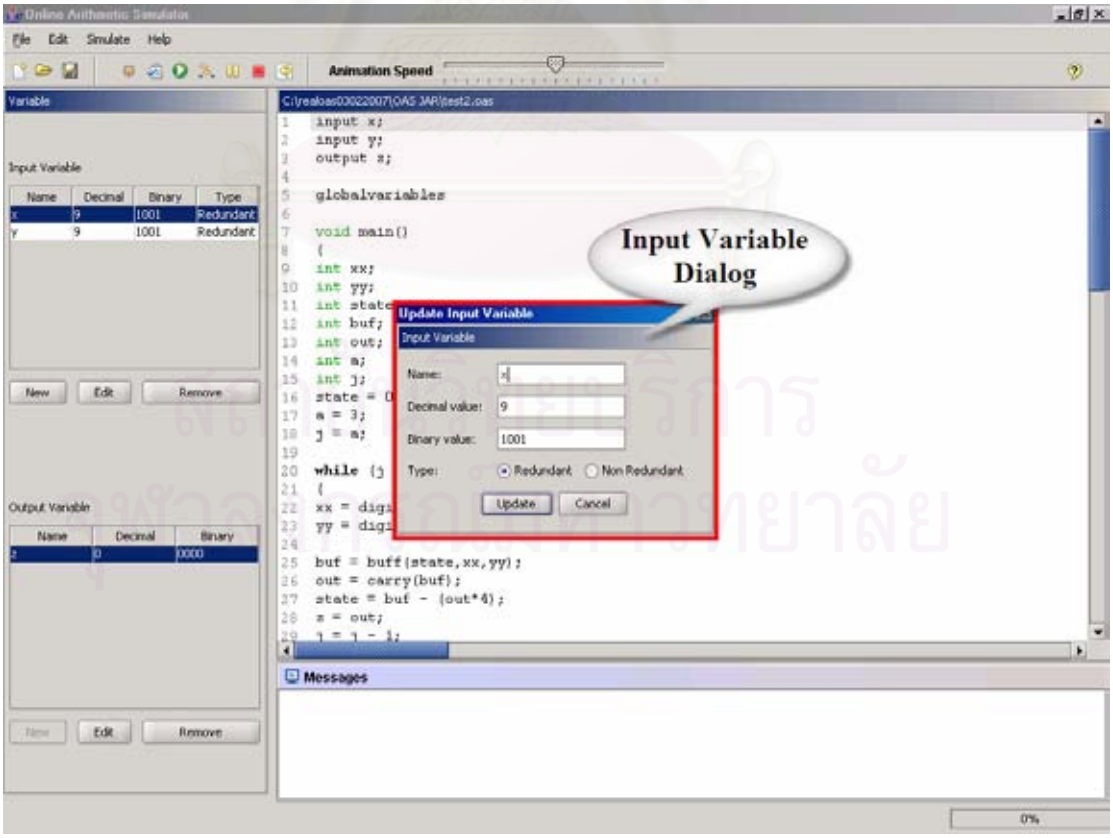
รูปที่ 5.8 ส่วนการเขียนโปรแกรมต้นฉบับ

5. ส่วนป้อนข้อมูลตัวแปรนำเข้าและตัวแปรผลลัพธ์ (Variable Panel) คือ ส่วนที่ใช้กำหนดตัวแปรนำเข้าและตัวแปรผลลัพธ์ที่ประกาศไว้ในโปรแกรมต้นฉบับ การใช้งานผู้ใช้เพิ่มตัวแปรนำเข้าได้ไม่เกิน 20 ตัว และตัวแปรผลลัพธ์ไม่เกินหนึ่งตัว หากมีตัวแปรในระบบถึงข้อกำหนดปุ่มที่ใช้เพิ่มตัวแปรจะถูกปิดทางเพื่อป้องกันการเพิ่มตัวแปรเกินข้อกำหนด ดังรูปที่ 5.9

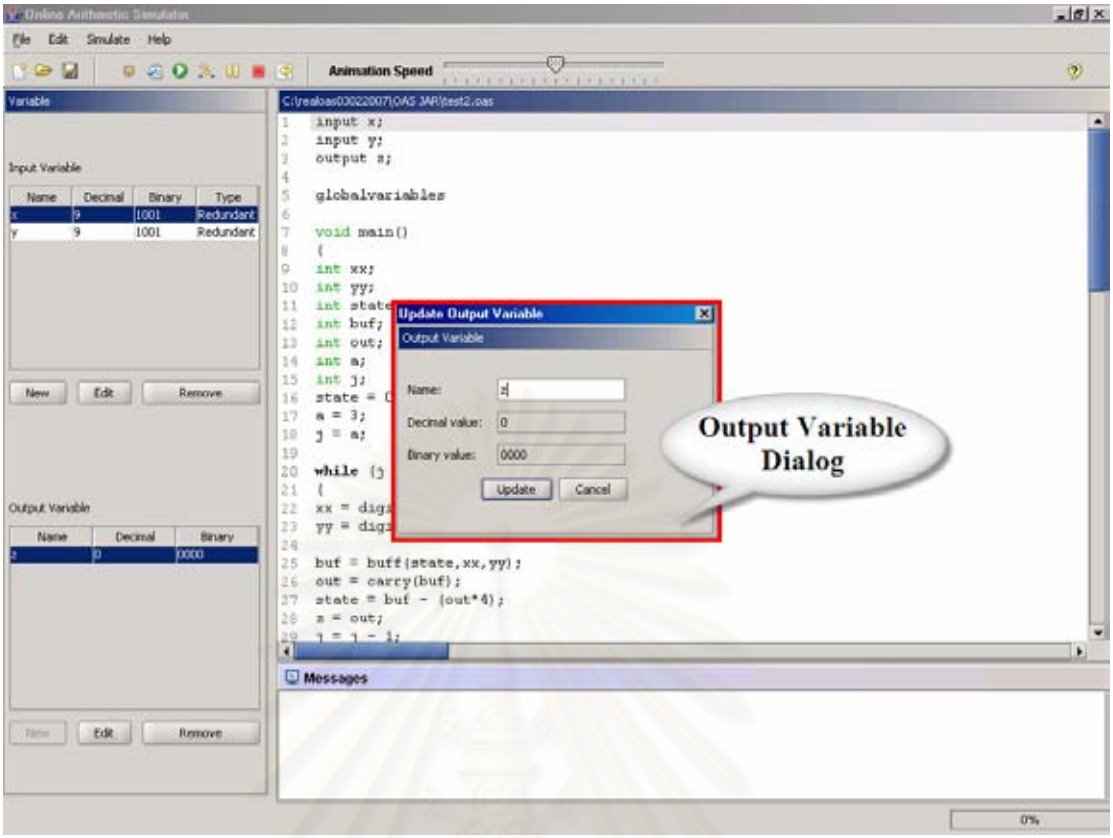
6. ส่วนโต้ตอบ (Variable Dialog) คือ ส่วนสำหรับเพิ่มและแก้ไขค่าของตัวแปรนำเข้าและตัวแปรผลลัพธ์ โดยจะทำหน้าที่ควบคุมความถูกต้องในระหว่างป้อนข้อมูล เช่น หากผู้ใช้พยายามป้อนตัวอักษรในช่อง “ค่าตัวเลขฐานสิบ” (Decimal value) หรือผู้ใช้ป้อนค่าที่ไม่ใช่ค่า -1, 0, 1 ในช่อง “ค่าของตัวเลขฐานสอง” (Binary value) ส่วนโต้ตอบจะไม่รับการป้อนค่าเหล่านั้น เป็นต้น ดังรูปที่ 5.10 - 5.11



รูปที่ 5.9 ส่วนป้อนข้อมูลตัวแปรนำเข้าและตัวแปรผลลัพธ์



รูปที่ 5.10 หน้าจอแก้ไขค่าตัวแปรนำเข้า

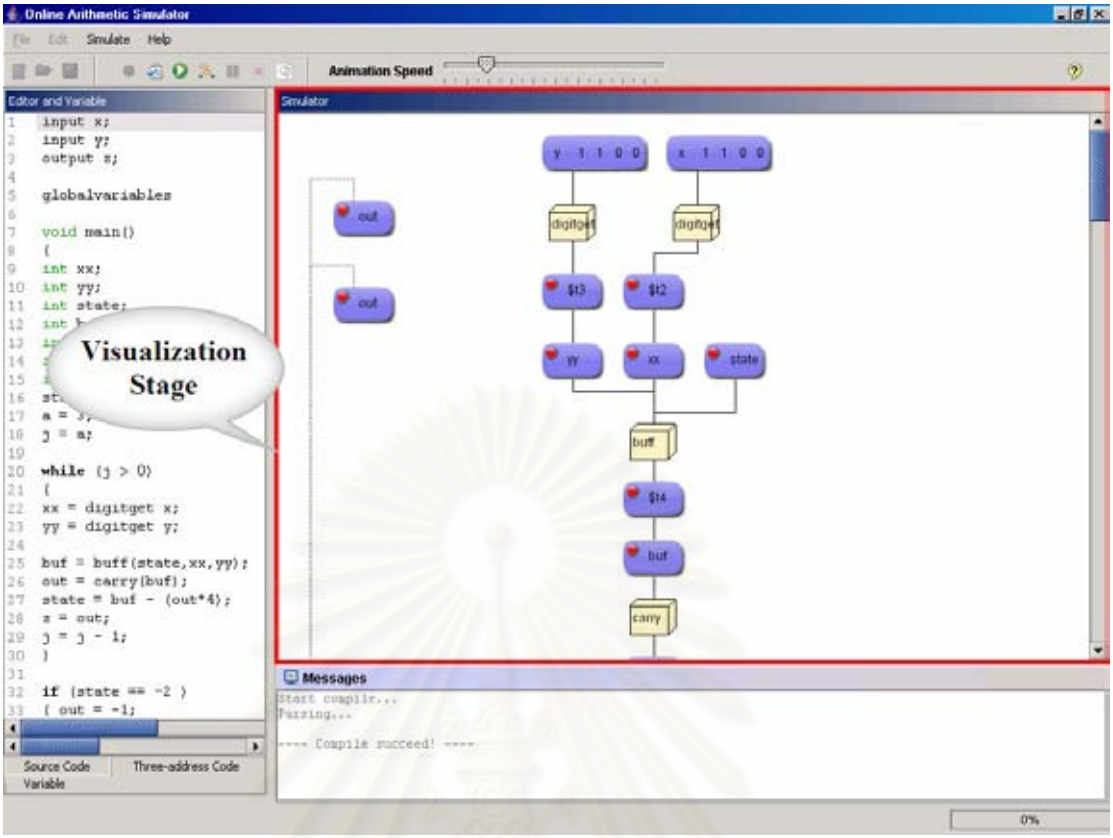


รูปที่ 5.11 หน้าจอแก้ไขค่าตัวแปรผลลัพธ์

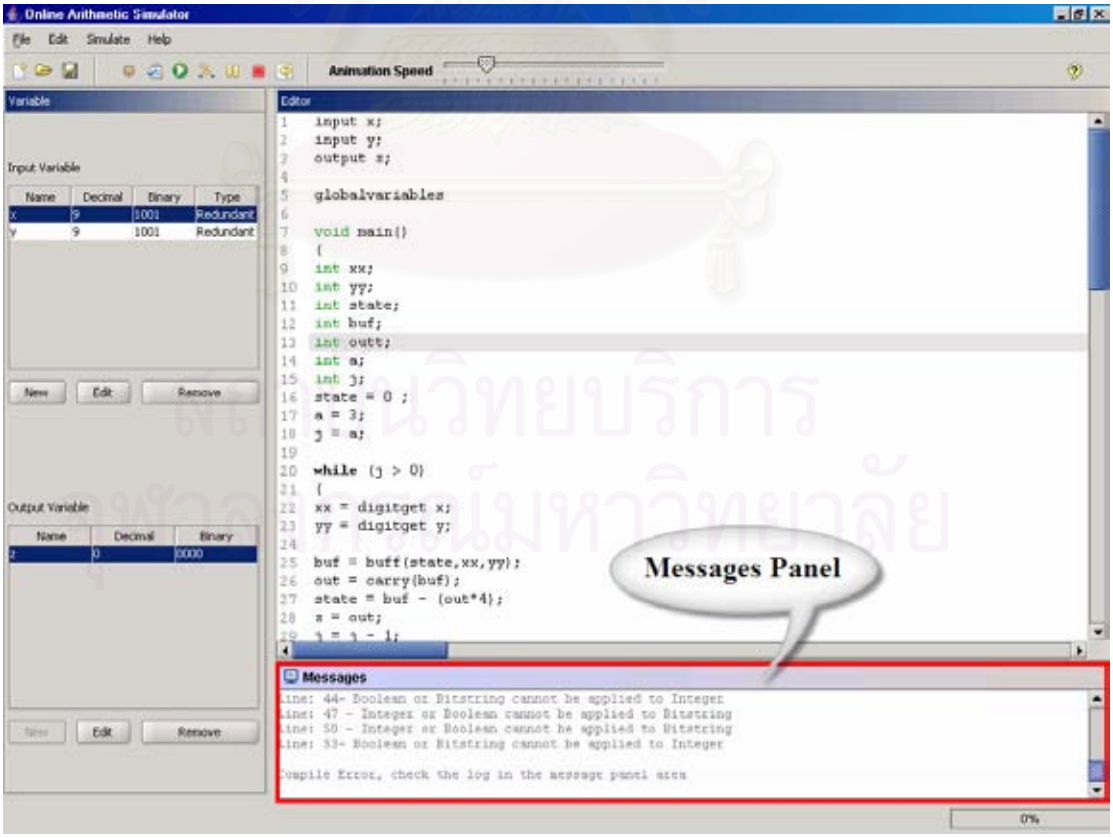
7. พื้นที่ส่วนแสดงภาพมโนทัศน์ (Animation Stage) คือ ส่วนสำหรับแสดงภาพมโนทัศน์การทำงานของอัลกอริทึม โดยจะแสดงผลแทนที่ส่วนการเขียนโปรแกรมต้นฉบับหากพบว่าการตรวจสอบการแปลภาษานั้นถูกต้อง ส่วนแสดงภาพมโนทัศน์จะถ่ายทอดคำสั่งให้แสดงผลการทำงาน ดังรูปที่ 5.12

8. ส่วนแสดงข้อความ (Messages Panel) คือ ส่วนใช้แสดงข้อความโต้ตอบกับผู้ใช้งาน โดยจะแสดงข้อความที่ได้รับมาจากส่วนแปลภาษาโอเอเอส เพื่อแสดงให้ผู้ใช้ทราบสถานะการตรวจสอบโปรแกรมต้นฉบับนั้นถูกต้องหรือไม่ นอกจากนี้ส่วนการแสดงข้อความยังแสดงข้อความถ้าพบข้อผิดพลาดในขณะที่กำลังแสดงภาพมโนทัศน์ ดังรูปที่ 5.13

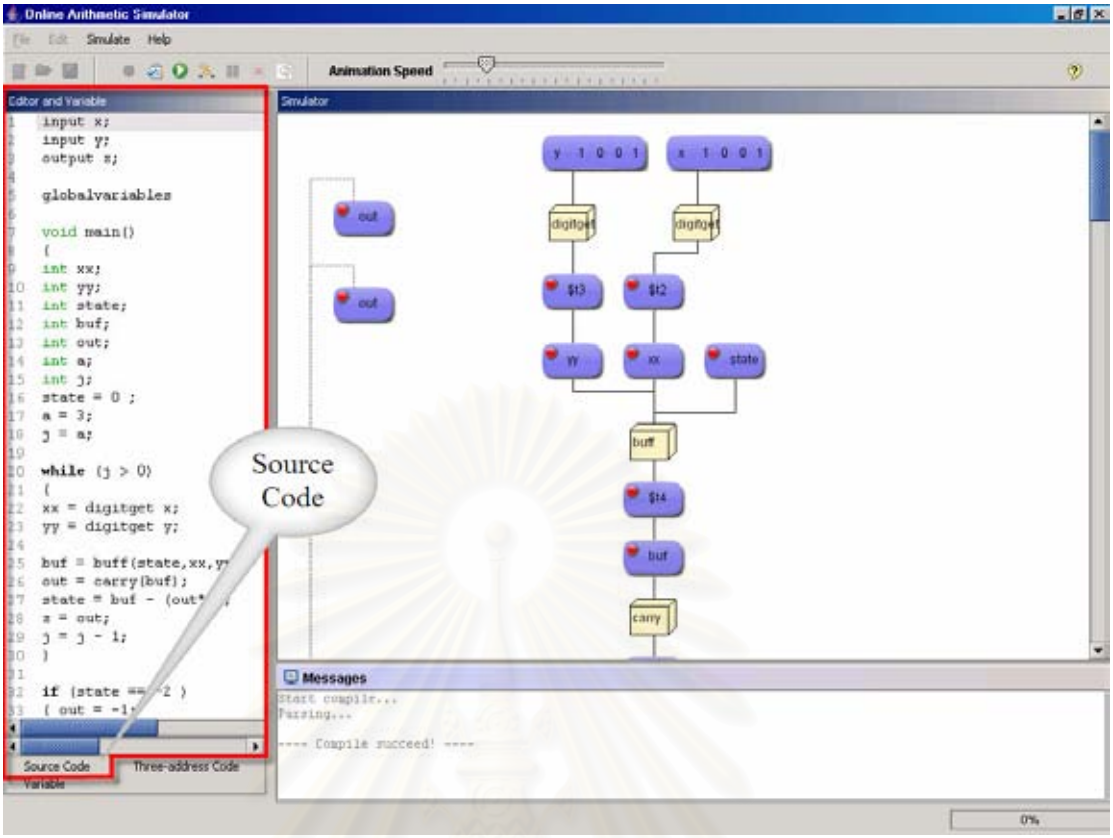
เมื่อเครื่องมืออยู่ในสถานะตรวจสอบความถูกต้องของโปรแกรมต้นฉบับเรียบร้อยแล้วจากตัวแปลภาษาโอเอเอส เครื่องมือจะเรียกใช้งาน TabPanel โดยโปรแกรมต้นฉบับจะถูกย้ายไปอยู่ทางด้านซ้ายมือของหน้าจอหลักภายในแท็บ “Source Code” ดังรูปที่ 5.14 พร้อมกับส่วนแสดงคำสั่งอินเตอร์มีเดียทที่ได้รับจากการแปลงโปรแกรมต้นฉบับแสดงภายในแท็บ “Three-address code” ดังรูปที่ 5.15 และแสดงแท็บ “Variable” แสดงค่าของตัวแปรนำเข้าและตัวแปรผลลัพธ์ โดยที่ค่าตัวแปรผลลัพธ์จะแสดงคำตอบที่ถูกผลิตออกมาจากการทำงานของอัลกอริทึม ดังรูปที่ 5.16



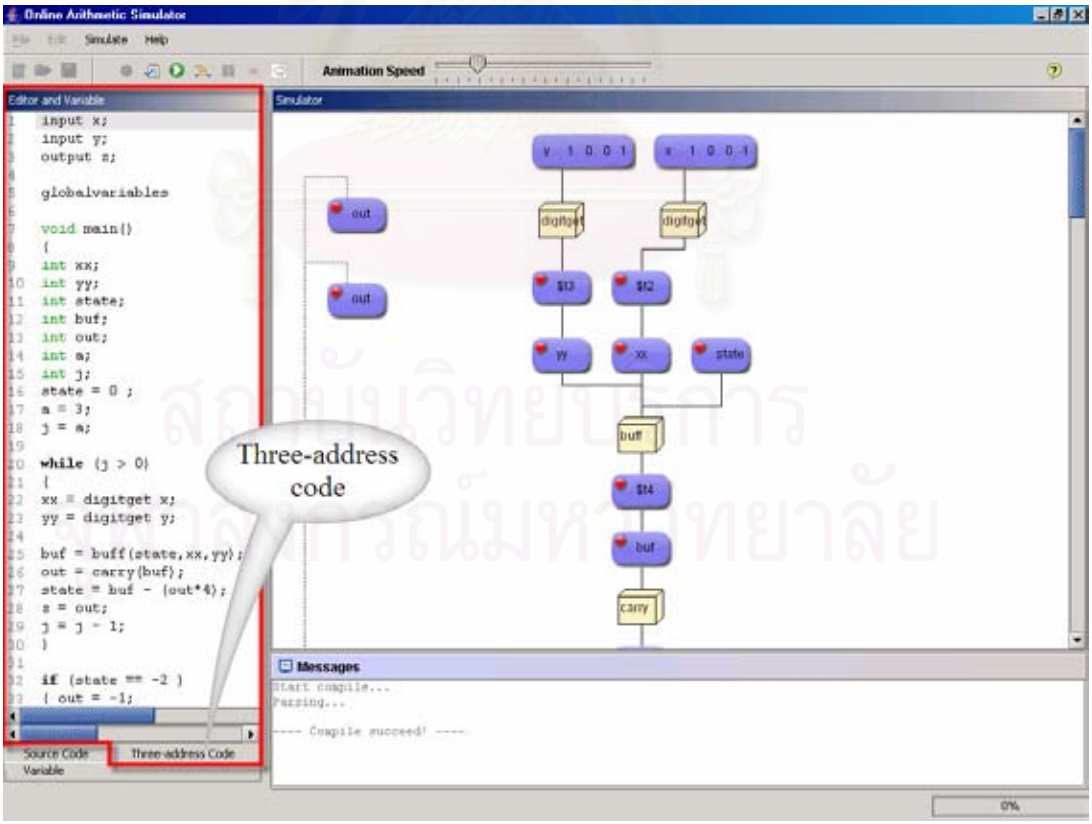
รูปที่ 5.12 พื้นที่ส่วนแสดงภาพมโนทัศน์



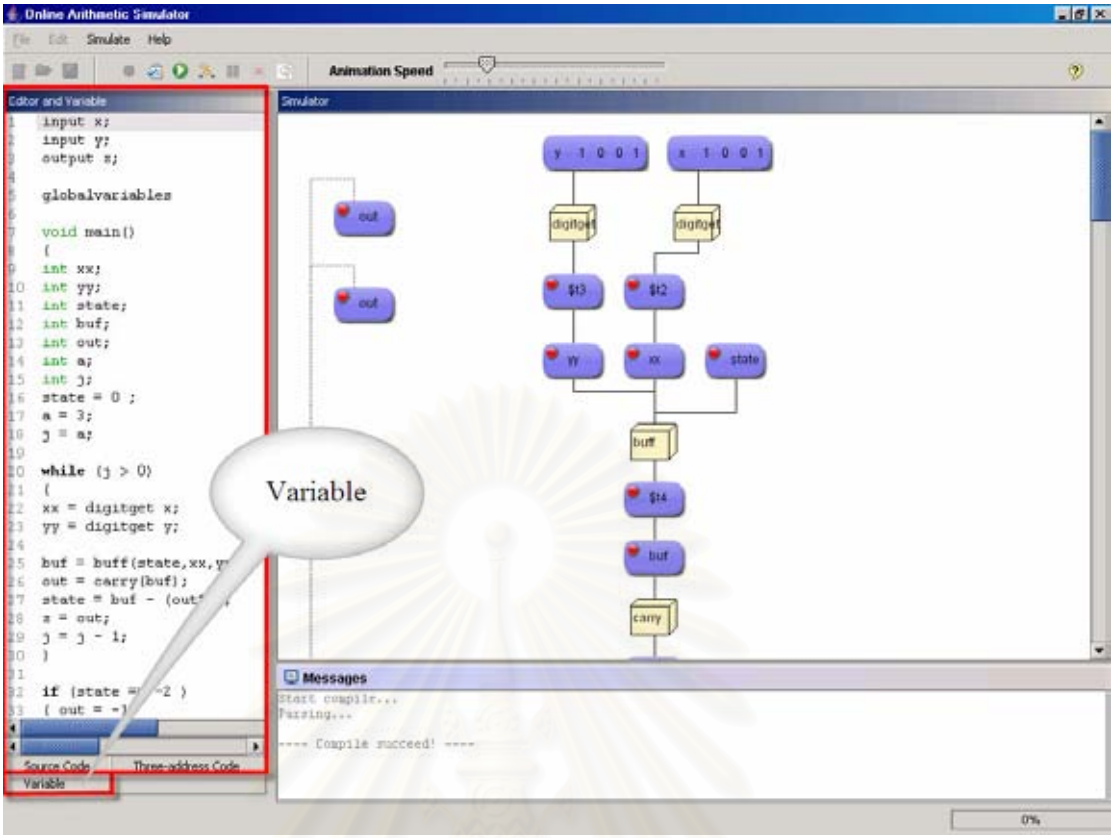
รูปที่ 5.13 พื้นที่ส่วนแสดงข้อความ



รูปที่ 5.14 แท็บ “Source Code”



รูปที่ 5.15 แท็บ “Three-address code”



รูปที่ 5.16 แท็บ “Variable”

5.3 การทดสอบ

ผู้จัดทำวิทยานิพนธ์ได้ทำการทดสอบการทำงานของเครื่องมือสร้างภาพมโนทัศน์สำหรับผู้ออกแบบอัลกอริทึมการคำนวณแบบเชื่อมตรงระดับดิจิต โดยกำหนดสภาพแวดล้อมที่ใช้ทดสอบ และกรณีทดสอบการทำงาน ประกอบด้วย การทดสอบการกำหนดตัวแปรนำเข้าและตัวแปรผลลัพธ์ การทดสอบตัวแปลภาษา การทดสอบการแปลคำสั่งอินเตอร์มิเดียท การทดสอบการสร้างภาพมโนทัศน์ การทดสอบการควบคุมการทำงานของการแสดงภาพมโนทัศน์ มีขั้นตอนการทำงาน ดังนี้

5.3.1 สภาพแวดล้อมที่ใช้ในการทดสอบ

เป็นสภาพแวดล้อมเดียวกันกับที่ใช้ในการพัฒนาเครื่องมือ

5.3.2 กรณีทดสอบ

5.3.2.1 การทดสอบการกำหนดตัวแปรนำเข้าและตัวแปรผลลัพธ์

เป็นการทดสอบการป้อนข้อมูลของตัวแปรนำเข้าและตัวแปรผลลัพธ์ โดยการป้อนชื่อ, ชนิดข้อมูล, การป้อนค่าตัวแปร โดยทดสอบตามลำดับขั้นตอน ดังนี้

ตารางที่ 5.1 กรณีทดสอบการกำหนดค่าตัวแปรนำเข้า

หมายเลขกรณีทดสอบ	OAS-1	
ชื่อ	กรณีทดสอบการกำหนดตัวแปรนำเข้า	
วัตถุประสงค์	เพื่อทดสอบการกำหนดตัวแปรนำเข้า,ชนิดข้อมูล,การป้อนค่าตัวแปร	
คำอธิบาย	1.กำหนดตัวแปรนำเข้า,ชนิดข้อมูล,การป้อนค่าตัวแปร 2.จัดเก็บในรายการของตัวแปรนำเข้าเพื่อนำไปใช้ในการคำนวณ	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถกำหนดข้อมูลตัวแปรนำเข้าและตรวจสอบความถูกต้องได้	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถกำหนดข้อมูลตัวแปรนำเข้าและตรวจสอบความถูกต้องได้	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.2 กรณีทดสอบการกำหนดค่าตัวแปรผลลัพธ์

หมายเลขกรณีทดสอบ	OAS-2	
ชื่อ	กรณีทดสอบการกำหนดตัวแปรผลลัพธ์	
วัตถุประสงค์	เพื่อทดสอบการกำหนดตัวแปรผลลัพธ์	
คำอธิบาย	1.กำหนดตัวแปรผลลัพธ์ 2.จัดเก็บตัวแปรผลลัพธ์สำหรับใช้ในการรับค่าการคำนวณ	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถกำหนดข้อมูลตัวแปรผลลัพธ์และตรวจสอบความถูกต้องได้	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถกำหนดข้อมูลตัวแปรผลลัพธ์และตรวจสอบความถูกต้องได้	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

5.3.2.2 การทดสอบตัวแปลภาษา

เป็นขั้นตอนการตรวจสอบความถูกต้องโปรแกรมต้นฉบับด้วยตัวแปลภาษา โดยทดสอบตามลำดับขั้นตอนของตัวแปลภาษา ดังนี้

ตารางที่ 5.3 กรณีทดสอบการตรวจสอบรูปแบบประโยคคำสั่ง

หมายเลขกรณีทดสอบ	OAS-3	
ชื่อ	กรณีทดสอบการตรวจสอบรูปแบบประโยคคำสั่ง	
วัตถุประสงค์	เพื่อทดสอบความถูกต้องของการตรวจสอบรูปแบบประโยคคำสั่งของโปรแกรมต้นฉบับ	
คำอธิบาย	1.ป้อนโปรแกรมต้นฉบับ 2.ตรวจสอบรูปแบบประโยคคำสั่ง เช่น คำสั่งประกาศ, คำสั่งกำหนดค่า, คำสั่งควบคุมเส้นทางการทำงาน	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	ตัวแปลภาษาสามารถตรวจสอบรูปแบบประโยคคำสั่งได้อย่างถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	ตัวแปลภาษาสามารถตรวจสอบรูปแบบประโยคคำสั่งได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.4 กรณีทดสอบการจัดการตารางสัญลักษณ์

หมายเลขกรณีทดสอบ	OAS-4	
ชื่อ	กรณีทดสอบการจัดการตารางสัญลักษณ์	
วัตถุประสงค์	เพื่อทดสอบการจัดการกับตารางสัญลักษณ์	
คำอธิบาย	1.ป้อนโปรแกรมต้นฉบับ 2.จัดเก็บชื่อ, ชนิด และกำหนดขอบเขตของตัวแปร	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	ตัวแปลภาษาสามารถจัดเก็บชื่อ, ชนิด และกำหนดขอบเขตของตัวแปรข้อมูลได้อย่างถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	ตัวแปลภาษาสามารถจัดเก็บชื่อ, ชนิด และกำหนดขอบเขตของตัวแปรข้อมูลได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.5 กรณีทดสอบการตรวจสอบความหมาย

หมายเลขกรณีทดสอบ	OAS-5	
ชื่อ	กรณีทดสอบการตรวจสอบความหมาย	
วัตถุประสงค์	เพื่อทดสอบการตรวจสอบความสอดคล้องของชนิดข้อมูล	
คำอธิบาย	1.ป้อนโปรแกรมต้นฉบับ 2.ตรวจสอบชนิดตัวแปรและความสอดคล้องของข้อมูล	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	ตัวแปลภาษาสามารถตรวจสอบชนิดตัวแปรและความสอดคล้องของข้อมูลได้อย่างถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	ตัวแปลภาษาสามารถตรวจสอบชนิดตัวแปรและความสอดคล้องของข้อมูลได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.6 กรณีทดสอบการแปลงโปรแกรมต้นฉบับเป็นชุดคำสั่งอินเตอร์มีเดียท

หมายเลขกรณีทดสอบ	OAS-6	
ชื่อ	กรณีทดสอบการแปลงโปรแกรมต้นฉบับเป็นชุดคำสั่งอินเตอร์มีเดียท	
วัตถุประสงค์	เพื่อทดสอบการแปลงโปรแกรมต้นฉบับเป็นชุดคำสั่งอินเตอร์มีเดียท	
คำอธิบาย	1.ป้อนโปรแกรมต้นฉบับ 2.แปลงโปรแกรมต้นฉบับเป็นชุดคำสั่งอินเตอร์มีเดียท	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	ตัวแปลภาษาสามารถแปลงคำสั่งอินเตอร์มีเดียทจากโปรแกรมต้นฉบับได้อย่างถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	ตัวแปลภาษาสามารถแปลงคำสั่งอินเตอร์มีเดียทจากโปรแกรมต้นฉบับได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.7 กรณีทดสอบลำดับการคำนวณของตัวดำเนินการต่างๆ

หมายเลขกรณีทดสอบ	OAS-7	
ชื่อ	กรณีทดสอบลำดับการคำนวณของตัวดำเนินการต่างๆ	
วัตถุประสงค์	เพื่อทดสอบลำดับการคำนวณของตัวดำเนินการต่างๆ	
คำอธิบาย	1. ป้อนโปรแกรมต้นฉบับ 2. ตรวจสอบลำดับการคำนวณของตัวดำเนินการ 3. แสดงผลลัพธ์การคำนวณของตัวดำเนินการทางส่วนประสานงานผู้ใช้	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถการคำนวณค่าตามลำดับการคำนวณได้ถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถการคำนวณค่าตามลำดับการคำนวณได้ถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

5.3.2.3 การทดสอบการแปลคำสั่งอินเตอร์มิเดียท

คำสั่งอินเตอร์มิเดียทจะถูกนำมาประมวลผล โดยขั้นตอนนี้จะทำการทดสอบการกำหนดความสัมพันธ์ของตัวแปรและคำสั่งที่เกี่ยวข้องกับตัวแปรนำเข้า และตัวแปรผลลัพธ์ การทดสอบคำนวณค่าของชุดคำสั่งอินเตอร์มิเดียท และการสร้างชุดคำสั่งสำหรับภาพมโนทัศน์ โดยทดสอบตามลำดับขั้นตอน ดังนี้

ตารางที่ 5.8 กรณีทดสอบการกำหนดความสัมพันธ์ของตัวแปรและคำสั่งที่เกี่ยวข้องกับตัวแปรนำเข้า และตัวแปรผลลัพธ์

หมายเลขกรณีทดสอบ	OAS-8
ชื่อ	กรณีทดสอบการกำหนดความสัมพันธ์ของตัวแปรและคำสั่งที่เกี่ยวข้องกับตัวแปรนำเข้า และตัวแปรผลลัพธ์
วัตถุประสงค์	เพื่อทดสอบการกำหนดความสัมพันธ์ของตัวแปรและคำสั่งที่เกี่ยวข้องกับตัวแปรนำเข้า และตัวแปรผลลัพธ์
คำอธิบาย	-
เงื่อนไขในการทดสอบ	-
ผลลัพธ์ที่คาดว่าจะได้รับ	ตัวแปลคำสั่งอินเตอร์มิเดียทสามารถกำหนดความสัมพันธ์ของตัวแปรและคำสั่งที่เกี่ยวข้องกับตัวแปรนำเข้า และตัวแปรผลลัพธ์ ได้

ตารางที่ 5.8 (ต่อ) กรณีทดสอบการกำหนดความสัมพันธ์ของตัวแปรและคำสั่งที่เกี่ยวข้องกับตัวแปรนำเข้า และตัวแปรผลลัพธ์

หมายเลขกรณีทดสอบ	OAS-8	
ผลลัพธ์ที่เกิดขึ้นจริง	ตัวแปลคำสั่งอินเตอร์มิเดียทสามารถกำหนดความสัมพันธ์ของตัวแปรและคำสั่งที่เกี่ยวข้องกับตัวแปรนำเข้า และตัวแปรผลลัพธ์ ได้	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.9 กรณีทดสอบการคำนวณค่าของชุดคำสั่งอินเตอร์มิเดียท

หมายเลขกรณีทดสอบ	OAS-9	
ชื่อ	กรณีทดสอบการคำนวณค่าของชุดคำสั่งอินเตอร์มิเดียท	
วัตถุประสงค์	เพื่อทดสอบการคำนวณค่าของชุดคำสั่งอินเตอร์มิเดียท	
คำอธิบาย	1.กำหนดชุดคำสั่ง 2.ใช้ตัวดำเนินการทุกประเภทนำมาทดสอบ	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	ตัวแปลคำสั่งอินเตอร์มิเดียทสามารถคำนวณค่าของชุดคำสั่งอินเตอร์มิเดียทได้อย่างถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	ตัวแปลคำสั่งอินเตอร์มิเดียทสามารถคำนวณค่าของชุดคำสั่งอินเตอร์มิเดียทได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.10 กรณีทดสอบการสร้างชุดคำสั่งสำหรับภาพมโนทัศน์

หมายเลขกรณีทดสอบ	OAS-10
ชื่อ	กรณีทดสอบการสร้างชุดคำสั่งสำหรับภาพมโนทัศน์
วัตถุประสงค์	เพื่อทดสอบการสร้างชุดคำสั่งสำหรับภาพมโนทัศน์
คำอธิบาย	-
เงื่อนไขในการทดสอบ	-
ผลลัพธ์ที่คาดว่าจะได้รับ	ตัวแปลคำสั่งอินเตอร์มิเดียทสามารถสร้างชุดคำสั่งสำหรับภาพมโนทัศน์ได้อย่างถูกต้อง

ตารางที่ 5.10 (ต่อ) กรณีทดสอบการสร้างชุดคำสั่งสำหรับภาพมโนทัศน์

หมายเลขกรณีทดสอบ	OAS-10	
ผลลัพธ์ที่เกิดขึ้นจริง	ตัวแปลคำสั่งอินเตอร์มิเดียทสามารถสร้างชุดคำสั่งสำหรับภาพมโนทัศน์ได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

5.3.2.4 การทดสอบการสร้างภาพมโนทัศน์

เป็นการทดสอบการแปลงโครงสร้างข้อมูลความสัมพันธ์เพื่อสร้างภาพมโนทัศน์และการสร้างหน่วยการแสดงที่เกี่ยวข้องกับชุดคำสั่งแสดงภาพมโนทัศน์ โดยทดสอบตามลำดับขั้นตอน ดังนี้

ตารางที่ 5.11 กรณีทดสอบการสร้างโครงสร้างข้อมูลสำหรับแสดงภาพมโนทัศน์

หมายเลขกรณีทดสอบ	OAS-11	
ชื่อ	กรณีทดสอบการแปลงโครงสร้างข้อมูลความสัมพันธ์เพื่อสร้างภาพมโนทัศน์	
วัตถุประสงค์	เพื่อทดสอบการแปลงโครงสร้างข้อมูลความสัมพันธ์เพื่อสร้างภาพมโนทัศน์	
คำอธิบาย	1. ป้อนโปรแกรมต้นฉบับ 2. แปลงคำสั่งอินเตอร์มีเดียทเป็นรูปภาพโครงสร้างข้อมูลความสัมพันธ์ของตัวแปร 3. แสดงรูปภาพทางส่วนประสานงานผู้ใช้	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถแปลงคำสั่งอินเตอร์มีเดียทเป็นรูปภาพโครงสร้างข้อมูลความสัมพันธ์ของตัวแปรได้	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถแปลงคำสั่งอินเตอร์มีเดียทเป็นรูปภาพโครงสร้างข้อมูลความสัมพันธ์ของตัวแปรได้	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.12 กรณีทดสอบการสร้างหน่วยการแสดงที่เกี่ยวข้องกับชุดคำสั่งแสดงภาพมโนทัศน์

หมายเลขกรณีทดสอบ	OAS-12	
ชื่อ	กรณีทดสอบการสร้างหน่วยการแสดงที่เกี่ยวข้องกับชุดคำสั่งแสดงภาพมโนทัศน์	
วัตถุประสงค์	เพื่อทดสอบการสร้างหน่วยการแสดงที่เกี่ยวข้องกับชุดคำสั่งแสดงภาพมโนทัศน์	
คำอธิบาย	1. ป้อนโปรแกรมต้นฉบับ 2. แปลงชุดคำสั่งแสดงภาพมโนทัศน์เป็นหน่วยของการแสดง 3. แสดงภาพเคลื่อนไหวผ่านส่วนประสานงานผู้ใช้	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถแปลงชุดคำสั่งแสดงภาพมโนทัศน์เป็นหน่วยของการแสดงได้	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถแปลงชุดคำสั่งแสดงภาพมโนทัศน์เป็นหน่วยของการแสดงได้	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.13 แสดงกรณีทดสอบการแสดงค่าเคลื่อนไหวในการรับ-ส่งข้อมูล

หมายเลขกรณีทดสอบ	OAS-13	
ชื่อ	กรณีทดสอบการแสดงค่าเคลื่อนไหวในการรับ-ส่งข้อมูล	
วัตถุประสงค์	ทดสอบการแสดงค่าเคลื่อนไหวในการรับ-ส่งข้อมูล	
คำอธิบาย	1. ป้อนโปรแกรมต้นฉบับ 2. แสดงรูปภาพลำดับการคำนวณทางส่วนประสานงานผู้ใช้ 3. แสดงผลลัพธ์เคลื่อนไหวในการรับ-ส่งข้อมูล	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถการคำนวณค่าตามลำดับการคำนวณได้ถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถการคำนวณค่าตามลำดับการคำนวณได้ถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.14 กรณีทดสอบการแสดงสถานะและการคำนวณการนำข้อมูลเข้าและออกจากหน่วยประกอบย่อย

หมายเลขกรณีทดสอบ	OAS-14	
ชื่อ	กรณีทดสอบการแสดงสถานะและการคำนวณการนำข้อมูลเข้าและออกจากหน่วยประกอบย่อย	
วัตถุประสงค์	ทดสอบการแสดงสถานะและการคำนวณการนำข้อมูลเข้าและออกจากหน่วยประกอบย่อย	
คำอธิบาย	แสดงสถานะและการคำนวณการนำข้อมูลเข้าและออกจากหน่วยประกอบย่อย	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถแสดงสถานะและการคำนวณการนำข้อมูลเข้าและออกจากหน่วยประกอบย่อย	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถแสดงสถานะและการคำนวณการนำข้อมูลเข้าและออกจากหน่วยประกอบย่อยได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

5.3.2.5 การทดสอบการควบคุมการทำงานของการแสดงภาพมโนทัศน์

เป็นการทดสอบการควบคุมการทำงานของการแสดงภาพมโนทัศน์ ผ่านส่วนประสานงานกับผู้ใช้ โดยทดสอบตามลำดับขั้นตอน ดังนี้

ตารางที่ 5.15 กรณีทดสอบการสั่งแสดงภาพมโนทัศน์

หมายเลขกรณีทดสอบ	OAS-15	
ชื่อ	กรณีทดสอบการสั่งแสดงภาพมโนทัศน์	
วัตถุประสงค์	ทดสอบการสั่งแสดงภาพมโนทัศน์	
คำอธิบาย	1. สั่งแสดงภาพมโนทัศน์การทำงาน 2. แสดงภาพมโนทัศน์ผ่านทางส่วนประสานงานผู้ใช้	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถแสดงภาพมโนทัศน์ได้อย่างถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถแสดงภาพมโนทัศน์ได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.16 กรณีทดสอบการสั่งหยุดแสดงภาพมโนทัศน์

หมายเลขกรณีทดสอบ	OAS-16	
ชื่อ	กรณีทดสอบการสั่งหยุดแสดงภาพมโนทัศน์	
วัตถุประสงค์	ทดสอบการสั่งหยุดแสดงภาพมโนทัศน์	
คำอธิบาย	1. สั่งหยุดแสดงภาพมโนทัศน์ 2. แสดงหยุดภาพมโนทัศน์ผ่านทางส่วนประสานงานผู้ใช้	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถแสดงการสั่งหยุดแสดงภาพมโนทัศน์ได้อย่างถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถแสดงการสั่งหยุดแสดงภาพมโนทัศน์ได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.17 กรณีทดสอบการสั่งหยุดแสดงภาพมโนทัศน์ชั่วคราว

หมายเลขกรณีทดสอบ	OAS-17	
ชื่อ	กรณีทดสอบการสั่งหยุดแสดงภาพมโนทัศน์ชั่วคราว	
วัตถุประสงค์	ทดสอบการสั่งหยุดแสดงภาพมโนทัศน์ชั่วคราว	
คำอธิบาย	1. สั่งหยุดภาพแสดงมโนทัศน์ชั่วคราว 2. แสดงสั่งหยุดภาพมโนทัศน์ชั่วคราวผ่านทางส่วนประสานงานผู้ใช้	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถแสดงการสั่งหยุดภาพมโนทัศน์ชั่วคราวได้อย่างถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถแสดงการสั่งหยุดภาพมโนทัศน์ชั่วคราวได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

ตารางที่ 5.18 กรณีทดสอบแสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่ง

หมายเลขกรณีทดสอบ	OAS-18	
ชื่อ	กรณีทดสอบแสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่ง	
วัตถุประสงค์	ทดสอบแสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่ง	
คำอธิบาย	1. สั่งแสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่ง 2. แสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่งผ่านทางส่วนประสานงานผู้ใช้	
เงื่อนไขในการทดสอบ	-	
ผลลัพธ์ที่คาดว่าจะได้รับ	สามารถแสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่งได้อย่างถูกต้อง	
ผลลัพธ์ที่เกิดขึ้นจริง	สามารถแสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่งได้อย่างถูกต้อง	
ผลการทดสอบ	☒ ผ่าน	☐ ไม่ผ่าน
หมายเหตุ	-	

5.3.3 สรุปผลการทดสอบ

สามารถสรุปผลการทดสอบจากที่ได้กล่าวข้างต้นทั้งหมด 18 กรณี ตามหมายเลขกรณีทดสอบได้ดังนี้

ตารางที่ 5.19 แสดงสรุปผลการทดสอบของแต่ละกรณี

ลำดับ	หมายเลขกรณีทดสอบ	ชื่อ	ผ่าน	ไม่ผ่าน
1	OAS-1	กรณีทดสอบการกำหนดค่าตัวแปรนำเข้า	☒	☐
2	OAS-2	กรณีทดสอบการกำหนดค่าตัวแปรผลลัพธ์	☒	☐
3	OAS-3	กรณีทดสอบการตรวจสอบรูปแบบประโยคคำสั่ง	☒	☐
4	OAS-4	กรณีทดสอบการจัดการตารางสัญลักษณ์	☒	☐
5	OAS-5	กรณีทดสอบการตรวจสอบความหมาย	☒	☐
6	OAS-6	กรณีทดสอบการแปลงโปรแกรมต้นฉบับเป็นชุดคำสั่งอินเตอร์มีเดียท	☒	☐
7	OAS-7	กรณีทดสอบลำดับการคำนวณของตัวดำเนินการต่างๆ	☒	☐

ตารางที่ 5.19 (ต่อ) แสดงสรุปผลการทดสอบของแต่ละกรณี

ลำดับ	หมายเลขกรณีทดสอบ	ชื่อ	ผ่าน	ไม่ผ่าน
8	OAS-8	กรณีทดสอบการกำหนดความสัมพันธ์ของตัวแปรและคำสั่งที่เกี่ยวข้องกับตัวแปรนำเข้า และตัวแปรผลลัพธ์	☒	☐
9	OAS-9	ทดสอบการคำนวณค่าของชุดคำสั่งอินเตอร์มิเดียท	☒	☐
10	OAS-10	กรณีทดสอบการสร้างชุดคำสั่งสำหรับภาพมโนทัศน์	☒	☐
11	OAS-11	ทดสอบการสร้างโครงสร้างข้อมูลสำหรับแสดงภาพมโนทัศน์	☒	☐
12	OAS-12	กรณีทดสอบการสร้างหน่วยการแสดงที่เกี่ยวข้องกับชุดคำสั่งแสดงภาพมโนทัศน์	☒	☐
13	OAS-13	แสดงกรณีทดสอบการแสดงค่าเคลื่อนไหวในการรับ-ส่งข้อมูล	☒	☐
14	OAS-14	กรณีทดสอบการแสดงสถานะและการคำนวณการนำข้อมูลเข้าและออกจากหน่วยประกอบย่อย	☒	☐
15	OAS-15	กรณีทดสอบการสั่งแสดงภาพมโนทัศน์	☒	☐
16	OAS-16	กรณีทดสอบการสั่งหยุดแสดงภาพมโนทัศน์	☒	☐
17	OAS-17	กรณีทดสอบการสั่งหยุดแสดงภาพมโนทัศน์ชั่วคราว	☒	☐
18	OAS-18	กรณีทดสอบแสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่ง	☒	☐

บทที่ 6

บทสรุปและข้อเสนอแนะ

6.1 บทสรุป

วิทยานิพนธ์ฉบับนี้ได้ทำการออกแบบและพัฒนาเครื่องมือสร้างภาพมโนทัศน์เพื่อสนับสนุนการออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิต โดยแสดงให้เห็นการเปลี่ยนแปลงเป็นรูปแผนภาพเคลื่อนไหว ที่สามารถช่วยให้ผู้ออกแบบประหยัดเวลาในการวางแผนผังของการประกอบหรือประหยัดเวลาในการออกแบบหน่วยย่อยของการคำนวณ ทำให้สามารถเห็นสถานะการทำงานของแต่ละหน่วยย่อยของการคำนวณสำหรับการออกแบบแผนผังวงจรที่ซับซ้อนได้ชัดเจนผ่านการสร้างภาพมโนทัศน์ก่อนการนำอัลกอริทึมไปสร้างวงจรการคำนวณจริง

การทำงานของเครื่องมือได้ออกแบบและพัฒนาตัวแปลภาษาและส่วนประมวลผลภาพมโนทัศน์ โดยการออกแบบภาษาสำหรับใช้แสดงภาพมโนทัศน์เพื่อนำมาใช้เขียนโปรแกรมด้วยอัลกอริทึมแบบเชื่อมตรง โดยแปลงโปรแกรมต้นฉบับให้อยู่ในรูปแบบชุดคำสั่งอินเตอร์มิเดียท ซึ่งเป็นรูปแบบคำสั่งเพื่อนำมาสร้างโครงสร้างข้อมูลและคำสั่งสำหรับแสดงภาพมโนทัศน์ ทำให้สามารถเห็นการส่งผ่านข้อมูลได้ชัดเจนในทุกขั้นตอนอยู่ในรูปแบบภาพกราฟิก ควบคุมการทำงานของภาพมโนทัศน์ผ่านทางส่วนประสานงานผู้ใช้

ผู้วิจัยได้ทำการทดสอบเครื่องมือทั้งในระหว่างการพัฒนา และภายหลังการพัฒนาเสร็จสิ้นแล้ว ผลที่ได้รับปรากฏว่าเครื่องมือสามารถทำงานได้อย่างถูกต้อง เป็นไปตามขั้นตอนตามที่ได้ออกแบบไว้

6.2 ปัญหาและข้อจำกัดของงานวิจัย

1) ตัวแปลภาษายังไม่สามารถทำการฟื้นสภาพข้อมูล (Recovery) ภายหลังจากพบข้อผิดพลาดในส่วนของการการวิเคราะห์คำ และการวิเคราะห์รูปประโยคได้ เมื่อพบข้อผิดพลาดจะทำให้เครื่องมือหยุดการทำงานทันที

2) ตัวแปรนำเข้าจำกัดอยู่เฉพาะการทำงานด้วยเลขฐานสองระดับดิจิตเท่านั้น

3) การแสดงผลภาพมโนทัศน์แสดงผลด้วยแฟ้มข้อมูลรูปภาพ ทำให้กรณีที่ข้อมูลมีความยาวเกินขนาดของรูปภาพไม่สามารถขยายขนาดตามความยาวของข้อมูลได้

4) เนื่องจากข้อจำกัดของการวาด การจัดตำแหน่งของแอคเตอร์ทำให้ผู้ใช้ไม่สามารถเคลื่อนย้ายแอคเตอร์อย่างอิสระหลังจากแสดงภาพมโนทัศน์โครงสร้างความสัมพันธ์ได้

5) หน่วยย่อยของการคำนวณแทนด้วยการเรียกใช้ฟังก์ชัน การแสดงภาพมโนทัศน์จะเห็นเป็นบล็อกของฟังก์ชันที่เรียกใช้ ซึ่งจะเห็นเฉพาะข้อมูลที่เข้าและออกจากฟังก์ชันเพื่อตรวจสอบค่าเท่านั้นยังไม่สามารถมองการทำงานด้านในฟังก์ชันได้

6.3 ข้อเสนอแนะ

หลังจากผู้วิจัยได้วิเคราะห์ ออกแบบ พัฒนา และทดสอบระบบจนเสร็จสมบูรณ์แล้ว ผู้วิจัยได้พบว่า ยังมีส่วนที่ต้องการพัฒนาเพิ่มเติมจากเครื่องมือสร้างภาพมโนทัศน์เพื่อสนับสนุนการออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิต ดังนี้

1) สร้างส่วนฟื้นสภาพข้อมูลภายหลังจากตรวจพบข้อผิดพลาด เพื่อทำให้สามารถตรวจสอบข้อผิดพลาดโดยรวมของโปรแกรมต้นฉบับได้

2) เปลี่ยนวิธีการสร้างชุดคำสั่งอินเตอร์มิเดียทให้สามารถสร้างและส่งคำสั่งให้ส่วนประมวลผลสร้างภาพมโนทัศน์ได้ โดยไม่จำเป็นต้องจัดเก็บลงในแฟ้มข้อมูลผลลัพธ์ก่อนนำไปประมวลผลการทำงาน

3) เพิ่มการสร้างส่วนขยายการแสดงรายละเอียดการทำงานภายในหน่วยย่อยการคำนวณหรือฟังก์ชันเพื่อให้สามารถเห็นความต่อเนื่องการทำงานที่ต้องการตรวจสอบได้

4) เปลี่ยนรูปแบบการวาดภาพมโนทัศน์โดยใช้เวคเตอร์กราฟิก เนื่องจาก การแสดงผลด้วยเวคเตอร์สามารถเปลี่ยนแปลงขนาดได้โดยไม่กระทบกับคุณภาพของภาพมโนทัศน์

5) ควรใช้ Drag and Drop API ในคลังโปรแกรม Swing ของ Java เข้ามาช่วยในการเคลื่อนย้ายภาพได้อย่างอิสระ เพื่อให้สามารถเปลี่ยนแปลงตำแหน่งของการแสดงภาพได้

6) ปรับปรุงขั้นตอนการสร้างชุดคำสั่งสำหรับแสดงภาพมโนทัศน์ให้ทำงานได้เร็วยิ่งขึ้น เนื่องจากเครื่องมือจะทำการสร้างชุดคำสั่งสำหรับแสดงภาพมโนทัศน์ทั้งหมดก่อนที่จะส่งให้ส่วนประมวลผลสร้างภาพมโนทัศน์ทำงาน

รายการอ้างอิง

[1] A. Avizienis, "Signed-digit number representation for fast parallel arithmetic", IRE Transactions on electronic Computer, Vol.10, 1961, PP. 389-400.

[2] M. D. Ercegovac, “On-line Arithmetic: an Overview SPIE 495”, *Real Time Signal Processing VII*, 1984, pp. 86-93.

[3] C. Demetrescu, I. Finocchi, and J. Stasko, “Specifying algorithm visualizations:Interesting events or state mapping?”, *Software Visualization: International Seminar*, 2001, pp. 16-30.

[4] Timothy Budd, "An Introduction to Object-Oriented Programming", Addison-Wesley, 2002.

[5] วีระศักดิ์ ซึงถาวร .*JAVA PROGRAMMING Volume II*. บริษัท ซีเอ็ดยูเคชั่น จำกัด (มหาชน),กรุงเทพฯ .2547

[6] A. V. Aho and R. Sethi and J. D. Ullman, “Compilers Principles Techniques and Tools”, Addison-Wesley, 1998.

[7] Kenneth C. Louden, “COMPILER CONSTRUCTION Principles and Practice”, PWS Publishing Inc, 1997.

[8] A. Appel: Modern Compiler Implementation in Java. Cambridge University Press, 1998

[9] M.H. Brown, “Algorithm animation”, Cambridge, MA : MIT Press, 1988.

[10] M.H. Brown, “Exploring algorithms using Balsa-II”, *IEEE Computer*, val. 21, 1988, pp. 14-36.

[11] ถาวร ลิขนะไพบูลย์. การออกแบบและพัฒนาระบบจินตทัศน์อัลกอริทึมสำหรับปัญหาทางทฤษฎีกราฟ.วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย. 2538

[12] ปวีณา ทองใบ. ระบบจินตทัศน์อัลกอริทึมของปัญหาทางด้านเรขาคณิตเชิงคำนวณ. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย. 2540

[13] ภูมิศักดิ์ วงศ์จงรุ่งเรื่อง. ระบบจินตทัศน์อัลกอริทึมการเรียงลำดับข้อมูล. วิทยานิพนธ์ปริญญามหาบัณฑิต ภาควิชาวิศวกรรมคอมพิวเตอร์ บัณฑิตวิทยาลัย จุฬาลงกรณ์มหาวิทยาลัย. 2538

[14] Jeliot 3. (online). Available from: http://cs.joensuu.fi/jeliot/index.php: 2004.

[15] Etienne Gagnon. SableCC, An Object-Oriented Compiler Framework. 1998. http://www.sablecc.org/downloads/thesis.pdf. Watched: 2004-05-13

[16] ผศ.ดร.สัลยุทธ์ สว่างวรรณ .คอมไพเลอร์. บริษัท วิรัตน์ เอ็ดดูเคชั่น จำกัด , กรุงเทพฯ .2547

[17] Jacquie Barker, "Beginning Java Objects From Concepts to Code, Second Edition", Apress, 2005.

[18] กิตติพงษ์ กลมกล่อม. DESIGN PATTERNS. บริษัท เคทีพี คอมพ์ แอนด์ คอนซัลท์ จำกัด. กรุงเทพฯ. 2548

[19] Mauro Marinilli, "Professional Java User Interfaces", John Wiley & Sons Ltd, 2006.

ภาคผนวก

ภาคผนวก ก
การบรรยายคำและบรรยายไวยากรณ์เพื่อสร้างตัวแปลภาษาโปรแกรมต้นฉบับด้วย SableCC

ภาคผนวกนี้จะกล่าวถึงการบรรยายคำและบรรยายไวยากรณ์ของโปรแกรมต้นฉบับลงในแฟ้มข้อมูลเฉพาะของเครื่องมือโครงสร้างการพัฒนาคอมไพเลอร์แบบเชิงวัตถุด้วย SableCC เพื่อนำมาใช้เป็นเครื่องมือสร้างตัวแปลภาษา ขั้นตอนการทำงานมีดังนี้

1. การบรรยายคำ เป็นการบรรยายด้วยข้อกำหนดในขั้นตอนการออกแบบภาษา จากตารางที่ ก-1 แบ่งการบรรยายคำในภาษาออกเป็น 4 รูปแบบ คือ อักขระคั่น (Delimiters) และตัวดำเนินการ, คำหลัก (Keywords), คำสั่งงานในตัว (Command built-in), หน่วยทางภาษา (Lexeme)

จากตารางที่ ก-1 ส่วนประกอบของการบรรยายคำสามารถอธิบายความหมายได้ดังนี้

- คำหลักของภาษาสามารถเขียนได้ทั้งอักษรตัวเล็กหรืออักษรตัวใหญ่ประกอบด้วยคำ ดังนี้

input	if	int
output	else	boolean
void	while	bitstring
return	globalvariables	xor
true	and	or
false	not	bitstrtoint
sbitstrtoint	inttobitstr	digitget

- การบรรยายสัญลักษณ์พิเศษ บรรทัดที่ 61 – 86 ประกอบด้วยสัญลักษณ์ดังต่อไปนี้

& | ^ ~ << >> >>> * + - / = < <= > >= == != ( ) { } [ ] ; , :=

- ชื่อตัวแปร, ตัวเลข, อักขระสายบิต บรรทัดที่ 91 – 93 บรรยายด้วย Regular Expression

ตารางที่ ก-1 การบรรยายคำโปรแกรมต้นฉบับ

```
1:  Helpers
2:  a = 'a' | 'A' ;
3:  b = 'b' | 'B' ;
4:  c = 'c' | 'C' ;
5:  d = 'd' | 'D' ;
6:  e = 'e' | 'E' ;
7:  f = 'f' | 'F' ;
```

ตารางที่ ก-1 (ต่อ) การบรรยายคำโปรแกรมต้นฉบับ

```
 8:  g = 'g' | 'G' ;
 9:  h = 'h' | 'H' ;
10:  i = 'i' | 'I' ;
11:   k = 'k' | 'K' ;
12:  l = 'l' | 'L' ;
13:  m = 'm' | 'M' ;
14:  n = 'n' | 'N' ;
15:  o = 'o' | 'O' ;
16:  p = 'p' | 'P' ;
17:  r = 'r' | 'R' ;
18:  s = 's' | 'S' ;
19:  t = 't' | 'T' ;
20:  u = 'u' | 'U' ;
21:  v = 'v' | 'V' ;
22:  w = 'w' | 'W' ;
23:  x = 'x' | 'X' ;
24:  y = 'y' | 'Y' ;
25:  letter = ['_' + [['a' .. 'z'] + ['A' .. 'Z']]];
26:  digit = ['0'..'9'];
27:  cr = 13 ;  // carriage return
28:  lf = 10 ;  // line feed
29:  tab = 9 ;  // tab char
30:  space = 32 ;  // space
31:  dquote = '"';
32:  escape = '\b' | '\t' | '\n' | '\f' | '\r' | '\"' | '\' "' | '\\';
33:  not_cr_lf = [tab + [[32..127] - [cr + lf]]];
34:  line_terminator = lf | cr | cr lf;
35:  not_star =    [not_cr_lf - '*'] | line_terminator;
36:  not_star_not_slash = [not_cr_lf - ['*' + '/']] | line_terminator;
37:  Tokens
38:  programs         = p r o g r a m s  ;
39:  globalvariables   = g l o b a l v a r i a b l e s ;
40:  input     = i n p u t  ;
41:  output    = o u t p u t ;
```

ตารางที่ ก-1 (ต่อ) การบรรยายคำโปรแกรมต้นฉบับ

```
42:  void     = v o i d ;
43:  boolean  = b o o l e a n ;
44:  int      = i n t ;
45:  bitstring = b i t s t r i n g ;
46:  bool     = ( t r u e | f a l s e) ;
47:  if       = i f ;
48:  else     = e l s e;
49:  while    = w h i l e;
50:  return   = r e t u r n ;
51:  f_bitstrtoint = b i t s t r t o i n t ;
52:  f_sbitstrtoint = s b i t s t r t o i n t ;
53:  f_inttobitstr = i n t t o b i t s t r ;
54:  f_inttosbitstr = i n t t o s b i t s t r ;
55:  f_get = d i g i t g e t ;
56:  and = a n d;
57:  or  = o r;
58:  not = n o t;
59:  xor = x o r ;


 /* bitwise Operators */
60:  bit_and = '&';
61:  bit_or = '|';
62:  bit_xor = '^';
63:  bit_complement = '~';
64:  bit_ls  = '<<' ;
65:  bit_rs  = '>>';
66:  bit_zfrs = '>>>';
 /* operators */
67:  mult = '*' ;
68:  plus = '+' ;
69:  minus = '-' ;
70  div = '/' ;
71:  assign = '=' ;
```

ตารางที่ ก-1 (ต่อ) การบรรยายคำโปรแกรมต้นฉบับ

```
72:   colonequal = ':=';
/* Relational operators */
73:   less = '<';
74:   lessequal = '<=';
75:   greater = '>';
76:   greaterequal = '>=';
 /* equality operators */
77:   equal = '==';
78:   notequal = '!=';
79:   l_parenthese = '(';
80:   r_parenthese = ')';
81:   l_brace = '{';
82:   r_brace = '}';
83:   l_bracket = '[';
84:   r_bracket = ']';
85:   semicolon = ';';
86:   comma = ',';
87:   new_line = cr | lf | cr lf;
88:   blank = ( tab | space )+;
89:   traditional_comment = '/*' not_star+ '*'+ (not_star_not_slash not_star* '*'+)* '/';
90:   comment = '//' not_cr_lf* line_terminator?;
91:   identifier       = letter ( letter | digit )* ;
92:   integer_literal   = digit+ ;
93:   bitstring_literal = dquote (['0' .. '1']+) dquote ;
94:   Ignored Tokens
95:   blank , new_line, traditional_comment, comment;
```

- อักขระที่ถูกละความสนใจ บรรทัดที่ 95 ประกอบด้วย อักขระตั้งระยะ (Tab), ตัวว่าง (Blank), อักขระขึ้นบรรทัดใหม่ (New lines) หมายเหตุประกอบด้วยเครื่องหมายบรรยายในบรรทัดที่ 89 – 90 โดยนำหน้าด้วย // หรือล้อมรอบด้วย /*…..*/

2. การบรรยายไวยากรณ์ (Grammar definition)

การบรรยายไวยากรณ์ลงในแฟ้มข้อมูลไวยากรณ์ SableCC เป็นส่วนของการกำหนดรูปแบบกฎเกณฑ์โครงสร้างของภาษา บรรยายในส่วนการประกาศ <Productions> โดยใช้อีบีเอ็นเอฟ [1] อักษรตัวหนาคือ สัญลักษณ์ที่รู้จัก ไม่ต้องตีความต่อไปอีกหรือที่เรียกว่า เทอร์มินอล (Terminal Symbol) อักษรตัวเอียงคือ ส่วนที่เป็นตัวแปรซึ่งจะต้องตีความต่อไปว่าใช้แทนโทเค้นใดบ้างหรือที่เรียกว่า นอนเทอร์มินอล (Non-terminals) แสดงดังตารางที่ ก-2

ตารางที่ ก-2 การบรรยายไวยากรณ์โปรแกรมต้นฉบับ

```
 1:  Productions
 2:  program  =
 3:  input_declaration*
 4:  output_declaration
 5:  global_identifier_block
 6:  globalvariables
 7:   function_declaration_block;
 8:  input_declaration  = input identifier semicolon ;
 9:  output_declaration = output identifier semicolon ;
10:  global_identifier_block          = global_identifier_declaration* ;
11:  global_identifier_declaration = identifier_declaration semicolon |
12:                                    {incorrect} identifier_declaration ;
13:  identifier_declaration =  {var_decl} var_declaration ;
14:  var_declaration = {unassigned} data_type identifier |
15:  {array} data_type identifier l_bracket integer_literal r_bracket ;
16:  array_access = l_bracket expression r_bracket;
17:   type_value =  condition;
18:   condition =
19:        {or}        condition or equality     |
20:        {xor}       condition xor equality    |
21:        {and}       condition and equality    |
22:        {bit_or}    condition bit_or equality |
23:        {bit_xor}   condition bit_xor equality |
24:        {bit_and}   condition bit_and equality |
25:        equality ;
26:   equality =
27:         {equal} equality equal relational |
```

ตารางที่ ก-2 (ต่อ) การบรรยายไวยากรณ์โปรแกรมต้นฉบับ

```
28:                 {equal_incorrect} equality assign relational |
29:                 {notequal} equality notequal relational |
30:                 relational ;
31:  relational =   {greater} relational greater expression |
32:                 {greaterequal} relational greaterequal expression |
33:                 {less} relational less expression |
34:                 {lessequal} relational lessequal expression |
35:                  bitop ;
36:  bitop       =  {zfrs}   bitop  bit_zfrs expression |
37:                 {bit_rs} bitop bit_rs expression |
38:                 {bit_ls} bitop bit_ls expression |
39:                 expression ;
40:  expression = {plus} expression plus term |
41:                 {minus} expression minus term |
42:                 {term} term ;
43:  term =         {mult} term mult not_operation |
44:                 {div} term div not_operation  |
45:                 not_operation ;
46:  not_operation = {not} not factor |
47:                   {bit_complement} bit_complement factor |
48:                    factor;
49:  factor = {condition} l_parenthese condition r_parenthese |
50:           {array} identifier array_access |
51:           {bool}  bool |
52:           value  ;
53:  value = {identifier} identifier |
54:           {integer_literal}  integer_literal |
55:          {signed_identifier} sign identifier |
56:          {signed_integer_literal} sign integer_literal |
57:          {bitstring_literal} bitstring_literal |
58:          {func_call_builtin} func_call_builtin |
59:          {function} function_call ;
60:  function_declaration_block =  function_declaration* ;
61:  function_declaration     =
```

ตารางที่ ก-2 (ต่อ) การบรรยายไวยากรณ์โปรแกรมต้นฉบับ

```
62:  function_return_value identifier l_parenthese declarator_parameters? r_parenthese
63:   function_block;
64:  declarator_parameters =  data_type identifier more_declarator_parameters? ;
65:  more_declarator_parameters =
66:   {correct} comma declarator_parameters |
67:   {incorrect} semicolon declarator_parameters ;
68:  function_block =
69:       {void_function} l_brace statement_list* r_brace |
70:       {return_function}   l_brace statement_list* return  expression semicolon r_brace;
71:  func_call_builtin =  {builtin_func_call} builtin_func_call ;
72:   builtin_func_call = {f_bitstrtoint}   f_bitstrtoint l_parenthese condition r_parenthese |
                          {f_sbitstrtoint}  f_sbitstrtoint l_parenthese condition r_parenthese |
                          {f_inttobitstr}   f_inttobitstr l_parenthese builtin_parameters? r_parenthese |
                          {f_inttosbitstr}  f_inttosbitstr l_parenthese builtin_parameters? r_parenthese |
                         {f_get} f_get identifier  ;
73:  builtin_parameters = builtin_call_value more_builtin_call_parameters?;
74:  more_builtin_call_parameters =   {correct}  comma builtin_parameters |
                                      {incorrect}   semicolon builtin_parameters   ;
75:  builtin_call_value = type_value;
76:  statement_list =   {block} block |  {while_loop} while_loop   |  {if_selection} if_selection   |
                       {statement} statement semicolon ;
77:   statement = {assignment} assignment |  {identifier_declaration} identifier_declaration |
     {function_call} function_call |  {empty} ;
78:   assignment = {correct}    variable assign  type_value |
                                {incorrect1} variable equal  type_value |
                                {incorrect2} variable colonequal type_value ;
79:   variable =  {array} identifier array_access |  {identifier} identifier;
80:   if_selection = if l_parenthese condition r_parenthese block else_selection? ;
81:   else_selection = else block ;
82:   while_loop = while l_parenthese condition r_parenthese block ;
83:   block = l_brace  statement_list* r_brace;
84:   function_call = identifier l_parenthese function_call_parameters? r_parenthese;
85:   function_call_parameters = function_call_value more_function_call_parameters?;
86:   more_function_call_parameters =  {correct} comma function_call_parameters |
```

ตารางที่ ก-2 (ต่อ) การบรรยายไวยากรณ์โปรแกรมต้นฉบับ

	{incorrect} **semicolon** *function_call_parameters* ;
87:	function_call_value = *type_value*;
88:	function_return_value = {data_type} *data_type* | {void} **void**;
89:	data_type = {int} **int** | {bitstring} **bitstring** | {boolean} **boolean** ;
90:	sign = {minus} **minus** ;

จากตารางที่ ก-2 เป็นการออกแบบไวยากรณ์ของภาษารายละเอียดมีดังนี้

บรรทัดที่ 2 คือ โปรดักชัน Program คือ โหนดรากของไวยากรณ์ประกอบด้วยลำดับของการบรรยายใน

บรรทัดที่ 3-4 คือ การบรรยายการประกาศตัวแปรนำเข้า, บรรยายการประกาศตัวแปรผลลัพธ์ โดยที่ตัวแปรนำเข้าสามารถประกาศได้มากกว่าหนึ่งตัว และตัวแปรผลลัพธ์สามารถประกาศได้เพียงหนึ่งตัวแปรเท่านั้น

บรรทัดที่ 5 คือ การบรรยายการประกาศตัวแปรส่วนกลาง

บรรทัดที่ 6 คือ คำหลัก globalvariables ใช้แบ่งการประกาศตัวแปรทั่วไปออกจากการประกาศตัวแปรแบบท้องถิ่นใช้ในการแก้ปัญหา reduce/reduce conflict [1]

บรรทัดที่ 7 คือ การบรรยายการประกาศฟังก์ชัน ทั้งหมดที่กล่าวถึงคือโครงสร้างของภาษา ส่วนต่อไปคือการบรรยายรายละเอียดส่วนที่เกี่ยวข้องมีดังนี้

บรรทัดที่ 10-16 คือ การบรรยายการประกาศตัวแปรและการกำหนดชนิดของตัวแปร

บรรทัดที่ 17-48 คือ การบรรยายการรูปแบบคำสั่งนิพจน์ที่ใช้กับตัวดำเนินการชนิดต่างๆ

บรรทัดที่ 49 คือ การบรรยายการรูปแบบนิพจน์เงื่อนไขสำหรับคำสั่งทางเลือก

บรรทัดที่ 53-57 คือ การบรรยายการรูปแบบของค่าที่ใช้กำหนดค่าให้กับตัวแปรประกอบด้วย ตัวแปร, ค่าคงที่, ฟังก์ชัน

บรรทัดที่ 61-70 คือ การบรรยายการรูปแบบการประกาศฟังก์ชันแบบส่งค่ากลับและแบบไม่ส่งค่ากลับ

บรรทัดที่ 71-74 คือ การบรรยายการรูปแบบการประกาศฟังก์ชันภายในและพารามิเตอร์

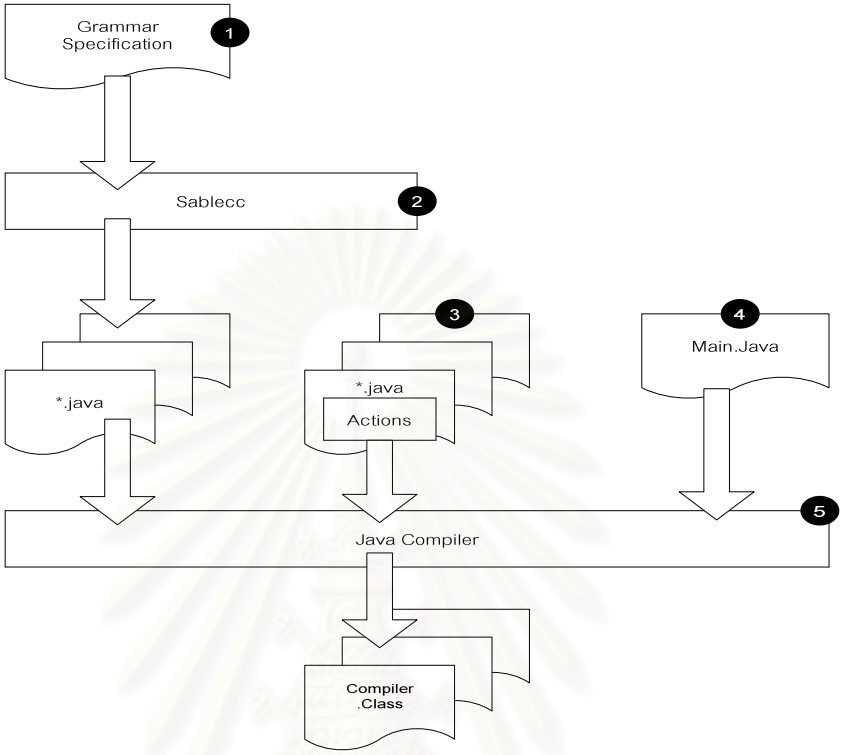
บรรทัดที่ 80-83 คือ การบรรยายการใช้คำสั่งต่างๆ ในภาษาแยกออกเป็นส่วนๆ ดังนี้

block คือ ส่วนการบรรยายชุดของประโยคคำสั่งภายใต้เครื่องหมายปีกกา ชุดประโยคคำสั่งประกอบด้วย

while_loop คือส่วนการบรรยายคำสั่งวนซ้ำรูปแบบแสดงในบรรทัดที่ 82 ประกอบด้วยโทเค็น while ตามด้วยประโยคเงื่อนไขภายใต้เครื่องหมายวงเล็บ ตามด้วยการบรรยายไวยากรณ์ของ block

if_selection คือ ส่วนการบรรยายคำสั่งทางเลือกแสดงรายละเอียดการบรรยายในบรรทัดที่ 80 คำสั่งประกอบด้วยโทเค็น if ตามด้วยประโยคเงื่อนไขภายใต้เครื่องหมายวงเล็บ ตามด้วยการบรรยายไวยากรณ์ของ block โดยอาจตามด้วยประโยคทางเลือก else หรือไม่ก็ได้ โดยที่การบรรยาย else จะใช้ else_selection? บรรยายรายละเอียดในบรรทัดที่ 81 รูปแบบคือ โทเค็น else ตามด้วยการบรรยายรูปแบบของ block แบบเรียกซ้ำ

3. แฟ้มข้อมูลเฉพาะ SableCC ใช้เป็นข้อมูลนำเข้าเพื่อใช้สร้างคลาสโครงร่างคอมไพเลอร์เชิงวัตถุ



รูปที่ ก-1 ขั้นตอนการพัฒนาคอมไพเลอร์เชิงวัตถุด้วย SableCC

3. คลาสทำงาน (Working Classes) สร้างโดยผู้พัฒนาตัวแปลภาษา โดยใช้การสืบทอดคุณสมบัติคลาสจากโครงสร้างตัวแปลภาษาเชิงวัตถุ

4. สร้างคลาส Main เพื่อควบคุมขั้นตอนการแปลภาษา

5. ตัวแปลภาษาจะถูกคอมไพล์ด้วยจาวาคอมไพเลอร์

Sablecc จะทำการสร้างคลาสโครงสร้างตัวแปลภาษาเชิงวัตถุ แบ่งออกเป็น 4 แพจเกจ (Package) คือ

เพจเกจ Lexer คือ เพจเกจที่ประกอบด้วยคลาสของตัววิเคราะห์คำ (Lexer Class) และคลาสของการตรวจจับข้อผิดพลาดของตัววิเคราะห์คำ (LexerException Class)

เพจเกจ Parser คือ เพจเกจที่ประกอบด้วยคลาสของการตรวจสอบรูปประโยค และคลาสของการตรวจจับข้อผิดพลาดของการตรวจสอบรูปประโยค (Parser Exception Classes) ขั้นตอนนี้จะรับเอาชุดของโทเก็นที่ส่งมาจากเพจเกจการตรวจสอบคำ เพื่อสร้างไวยากรณ์ต้นไม้นามธรรม

เพจเกจ Nodes คือ เพจเกจที่ประกอบด้วยโหนดคลาสทั้งหมดสำหรับใช้เป็นส่วนประกอบไวยากรณ์ต้นไม้นามธรรมประกอบด้วยคลาสโทเค็น (Token Class) นำหน้าด้วยตัวอักขระ T ลงท้ายด้วยชื่อโทเค็น คลาสโปรดักชัน (Production Class) นำหน้าด้วยตัวอักขระ P ลงท้ายด้วยชื่อโปรดักชัน คลาสโปรดักชัน (Production Class) สามารถมีได้มากกว่าหนึ่งคลาสทางเลือก (Alternative Class) นำหน้าด้วยตัวอักขระ A ลงท้ายด้วยชื่อ โปรดักชันอยู่ภายในเครื่องหมายปีกกา ตัวอย่างดังตารางที่ ก-3

ตารางที่ ก-3 การบรรยายไวยากรณ์นิพจน์ตัวดำเนินการทางคณิตศาสตร์

```
Expression =
{number} number |
{plus} expression plus number |
{minus} expression minus number ;
```

จากการกำหนดไวยากรณ์ดังตารางที่ ก-3 SableCC จะทำการการสร้างคลาสสำหรับใช้ในการกำหนดไวยากร์ต้นไม้ดังนี้

- โทเค็นคลาส คือ TNumber, TPlus, TMinus
- โปรดักชันคลาส คือ PExpression
- คลาสทางเลือก คือ ANumberExpression, APlusExpression และ AminusExpression

แพจเกจ Analysis คือ แพจเกจเก็บคลาสท่องโหนดไวยากรณ์ต้นไม้ (Tree Walkers) สำหรับใช้เยี่ยมทุกโหนดในไวยากรณ์ต้นไม้แบบกำหนดลำดับด้วยคลาส DepthFirstAdapter สำหรับใช้ท่องโหนดไวยากรณ์ต้นไม้ตามแนวลึก (Depth-First Traversal)

ภาคผนวก ข
การบรรยายคำและบรรยายไวยากรณ์เพื่อสร้างตัวแปลชุดคำสั่งอินเตอร์มิเดียทด้วย Grammatica

ภาคผนวกนี้จะกล่าวถึงการสร้างใช้การบรรยายคำ และไวยากรณ์ของคำสั่งอินเตอร์มิเดียท ดังตารางที่ ข-1 เพื่อสร้างคลาสตรวจสอบไวยากรณ์คำสั่งอินเตอร์มีเดียท

ตารางที่ ข-1 การบรรยายคำชุดคำสั่งอินเตอร์มิเดียท

```
Token part
INPUT = "input"
OUTPUT = "output"
NUM = <<[0-9]+>>
BIT = <<\"([-]*[0-1]*)*\">>

COMMA = ","
COLON = ":"
DIM = "dim"
ARRAY = "array"
INTEGER = "integer"
BOOLEAN = "boolean"
BITSTRING = "bitstring"

PLUS = "+"
MINUS = "-"
MUL = "*"
DIV = "/"

EQ = "="
GREATER = ">"
LESSER = "<"
GREATEQ = ">="
LESSEQ = "<="
EQUAL = "=="
NOTEQ = "!="
NOT = "not"
AND = "and"
```

ตารางที่ ข-1 (ต่อ) การบรรยายคำชุดคำสั่งอินเตอร์มิเดียท

```
OR = "or"
XOR = "xor"

NOTBIT = "~"
ANDBIT = "&"
ORBIT = "|"
XORBIT = "^"
SHIFTL = "<<"
SHIFTR = ">>"
USHIFTR = ">>>"

BTRUE = "True"
STRUE = "true"
BFALSE = "False"
SFALSE = "false"
WHITESPACE = <<[\s]+>> %ignore%
IF = "if"
GOTO = "GOTO"
UNDER = "_"
SHARP = "#"
LPAR = "("
RPAR = ")"
GET = "digitget"
MAIN = "main"
GLOBAL = "global"
STR = <<[a-zA-Z][0-9a-zA-Z]*>>
AT = "@"
DOLLAR = "$"

CALL = "call"
PARAM = "param"
RETURN = "return"
```

ตารางที่ ข-1 (ต่อ) การบรรยายคำชุดคำสั่งอินเตอร์มิเดียท

BITTOINT = "bitstrtoint"
INTTOBIT = "inttobitstr"
BITADD = "bitadd"
BITSUB = "bitsub"
SBITTOINT = "sbitstrtoint"
INTTOSBIT = "inttosbitstr"

จากตารางที่ ข-1 การสร้างโทเค็นขึ้นมาเพื่อใช้แทนคำที่ถูกใช้ในคำสั่งอินเตอร์มิเดียท โดยโทเค็นเหล่านี้จะทำให้โปรแกรมรู้จักว่าในหนึ่งบรรทัดของภาษาอินเตอร์มิเดียทมีคำศัพท์ใดประกอบขึ้นมาบ้างไวยากรณ์ด้านบนจะแบ่งออกเป็น 3 ประเภทได้แก่

1. โทเค็นที่แทนคำเฉพาะ คือ โทเค็นที่ใช้แทนภาษาอินเตอร์มิเดียทที่เป็นคำที่มีความหมายในตัวของมันเอง

- ชื่อฟังก์ชัน “bitstrtoint”, “main”, “global”, “dim”, “digitget” ค่าคงที่ “true”, “false”
- ชนิดของตัวแปร “integer”, “array”, “bitstring”

2. โทเค็นที่แทนเครื่องหมายต่างๆ ที่ใช้ในภาษาอินเตอร์มิเดียท

เครื่องหมายสำหรับการคำนวณตัวเลข “+”, “-“, “*”, “/”

- เครื่องหมายสำหรับค่าตรรกะ “and”, “or”, “>”, “==”
- เครื่องหมายสำหรับค่าตรรกะของ bitstring “&”, “|”, “~”
- เครื่องหมายอื่นๆ “$”, “@”, “#”, “(“, “)”

3. โทเค็นที่ใช้ Regular Expression เข้ามาเพื่อกำหนดช่วงหรือรูปแบบของตัวมันเอง

- ค่าของตัวเลข “NUM” กำหนดค่าไว้ในช่วงตั้งแต่ 0 ถึง
- ค่าของสายอักขระ bitstring “BIT” กำหนดค่าไว้ว่าจะต้องประกอบด้วย 0 หรือ 1 เท่านั้น

- ค่าของ string “STR” กำหนดไว้ว่าจะต้องประกอบไปด้วยตัวอักษร a ถึง z โดยที่จะเป็นตัวใหญ่หรือตัวเล็กก็ได้ หรือจะประกอบไปด้วยตัวเลขก็ได้ แต่ต้องขึ้นต้นด้วยตัวอักษรเท่านั้น ส่วนใหญ่จะใช้สำหรับการตั้งชื่อของตัวแปร

ตารางที่ ข-2 การบรรยายไวยากรณ์ชุดคำสั่งอินเตอร์มิเดียท

```
Production part
SENT = INPUTLINE | OUTPUTLINE | EQUATION |
                    CONDITION | MAINFUNC | FUNCTION | DECLARE | GOTOCOM | LINELABEL |
                    CALLFUNC | PARAMLINE | RETURNLINE;
VAR = (MAIN | GLOBAL | STR) UNDER ((
          (INTEGER | BITSTRING| BOOLEAN) UNDER [ARRAY UNDER] STR
[SHARP (NUM | VAR | TEMPVAR)]) | STR);
TEMPVAR = DOLLAR STR;
TYPE = INTEGER | BOOLEAN | BITSTRING;
ATOM = NT | MT | PARENT | BTRUE | STRUE | BFALSE | SFALSE | BITNUM;
MT = [MINUS] NUM;
NT = ([NOT | NOTBIT | MINUS] VAR) | ([NOT | NOTBIT | MINUS] TEMPVAR) | (NOT ((BTRUE |
         STRUE) | (BFALSE | SFALSE)));

PARENT = LPAR MT RPAR;
BITNUM = [NOTBIT] BIT;
GETTER = GET STR;
OPERATOR = ATOM [(PLUS | MINUS | MUL | DIV
          | GREATER | LESSER | GREATEQ | LESSEQ | NOTEQ | EQUAL | AND | OR | XOR
          | ANDBIT | ORBIT | XORBIT| SHIFTL | SHIFTR | USHIFTR) ATOM];
DECLARE = DIM VAR;
MAINFUNC = MAIN COLON;
FUNCTION = STR COLON;
LINELABEL = AT NUM COLON;
GOTOCOM = GOTO LINELABEL;
EQUATION = ATOM EQ (OPERATOR | GETTER | RETURNPARAMOBJ | BITTOINTCOM |
INTTOBITCOM |   BITADDCOM | BITSUBCOM | SBITTOINTCOM | INTTOSBITCOM);
```

ตารางที่ ข-2 (ต่อ) การบรรยายไวยากรณ์ชุดคำสั่งอินเตอร์มิเดียท

```
CONDITION = IF ATOM EQ (BTRUE | STRUE) GOTOCOM [GOTOCOM];
INPUTLINE = INPUT STR;
OUTPUTLINE = OUTPUT STR;
CALLFUNC = CALL STR COMMA NUM;
PARAMLINE = PARAM OPERATOR;
RETURNLINE = RETURN OPERATOR;
RETURNPARAMOBJ = SHARP (STR | RETURN);
BITTOINTCOM = BITTOINT LPAR (BITNUM | NT) RPAR;
INTTOBITCOM = INTTOBIT LPAR (NT | MT) COMMA (NUM | NT) RPAR;
BITSUBCOM = BITSUB LPAR (BITNUM | NT) COMMA (BITNUM | NT) RPAR;
SBITTOINTCOM = SBITTOINT LPAR (BITNUM | NT) RPAR;
INTTOSBITCOM = INTTOSBIT LPAR (NT | MT) COMMA (NUM | NT) RPAR;
```

Production คือ ส่วนของการบรรยายไวยากณ์คำสั่งอินเตอร์มิเดียทประกอบด้วยกลุ่มของโทเค็นเรียงตัวเป็นรูปแบบประโยค ดังนี้

“SENT” คือรากของโปรดักชันทุกตัว ซึ่งจะเป็นตัวบอกว่าบรรทัดหนึ่งบรรทัดในภาษาอินเตอร์มิเดียจสามารถที่จะเป็นรูปแบบอะไรได้บ้าง ดังนี้

- “INPUTLINE” คือบรรทัดที่บอกว่า ตัวแปรอินพุทชื่อว่าอะไร
- “OUTPUTLINE” คือบรรทัดที่บอกว่า ตัวแปรผลลัพธ์ชื่อว่าอะไร
- “EQUATION” คือบรรทัดสำหรับการกำหนดค่าให้ตัวแปร
- “CONDITION” คือบรรทัดสำหรับประโยคเงื่อนไข
- “MAINFUNC” คือบรรทัดเริ่มต้นของ Main function และทำให้รู้ว่าคำสั่งบรรทัดต่อๆไปจะอยู่ภายใต้ฟังก์ชันนี้
- “FUNCTION” คือบรรทัดเริ่มต้นของฟังก์ชันที่ผู้ใช้ประกาศขึ้นมาใช้เองและทำให้รู้ว่าคำสั่งบรรทัดต่อๆไปจะอยู่ภายใต้ฟังก์ชันนี้
- “DECLARE” คือบรรทัดของการประกาศตัวแปร
- “GOTOCOM” คือบรรทัดของการสั่งให้โปรแกรมย้ายการประมวลผลไปยังกลุ่มของโค้ดที่กำหนด

- “LINELABEL” คือบรรทัดเริ่มต้นของกลุ่มโค้ดและทำให้รู้ว่าคำสั่งบรรทัดต่อๆไปจะอยู่ภายใต้กลุ่มโค้ดนี้
- “CALLFUNC” คือบรรทัดของการเรียกใช้ฟังก์ชันที่ผู้ใช้ประกาศขึ้นมาทำงาน
- “PARAMLINE” คือบรรทัดที่เป็นตัวบอกว่าพารามิเตอร์ของฟังก์ชันที่กำลังจะเรียกคืออะไร
- “RETURNLINE” คือบรรทัดของการส่งผลลัพธ์หรือค่ากลับมาจากฟังก์ชันที่ถูกเรียกไปก่อนหน้านี้

“EQUATION” คือบรรทัดที่มีหน้าที่ในการกำหนดค่าให้ตัวแปรที่อยู่ด้านซ้ายของเครื่องหมายเท่ากับ หรือโทเค็น “EQ” ประกอบด้วย 2 ส่วนหลักๆคือ

- โปรดักชัน “ATOM” ที่อยู่ด้านซ้ายของโทเค็น “EQ” ภายในบรรทัดนี้โปรดักชันนี้จะถูกใช้แทนชื่อตัวแปร ซึ่งเป็นได้ทั้งตัวแปรผลลัพธ์, ตัวแปรทั่วไปหรือตัวแปรชั่วคราว เพื่อรับค่าจากโปรดักชันด้านขวาของเครื่องหมายเท่ากับมาเก็บไว้เพื่อใช้งานต่อไป
- โปรดักชันที่อยู่ด้านขวา เป็นโปรดักชันที่มีการส่งค่าออกมา โดยสามารถเป็นโปรดักชันได้หลายชนิด

โปรดักชัน “OPERATOR” เป็นโปรดักชันสำหรับการคำนวณค่าซึ่งอาจจะเป็นการคำนวณค่าของตัวเลข, ตรรกะ หรือ bitstring ก็ได้

โปรดักชันของ Built in function ต่างๆที่มีใช้ในภาษาอินเตอร์มิเดียท

“ATOM” คือโปรดักชันที่แทนค่าของตัวเลข, ตัวแปร, ค่าคงที่ตรรกกะ และ bitstring

“VAR” คือโปรดักชันที่แทนชื่อของตัวแปร ตัวแปรในภาษาอินเตอร์มิเดียทสามารถเป็นได้ 3 แบบคือ

- ตัวแปรที่ไม่มีตัวระบุตำแหน่ง ตัวแปรจะแบ่งออกเป็น 3 ส่วน คือ

  -ชื่อฟังก์ชันที่ตัวแปรตัวนี้อยู่

  -ชื่อประเภทของตัวแปร (integer, Boolean, bitstring)

  -ชื่อของตัวแปร

โดยแต่ละส่วนจะมีโทเค็น “UNDER” (_) เป็นตัวแบ่งส่วนเช่น main_integer_xx

- ตัวแปร bitstring แบบระบุตำแหน่ง แบ่งออกเป็น 5 ส่วนคือ

- ชื่อฟังก์ชันขอบเขตของตัวแปร
- ชื่อประเภทของตัวแปร (bitstring)
- ชื่อตัวแปร
- เครื่องหมาย “#”
- ตำแหน่งของตัวแปร

โดยแต่ละส่วนจะมีโทเค็น “UNDER” (_) เป็นตัวแบ่งส่วนเช่น main_bitstring_bb#1

- ตัวแปรอะเรย์ เป็นตัวแปรที่มีตำแหน่งระบุและคำว่า “array” ตัวแปรจะแบ่งออกเป็น 6 ส่วน คือ
  - ชื่อฟังก์ชันขอบเขตของตัวแปร
  - ชื่อประเภทของตัวแปร (integer, Boolean, bitstring)
  - คำว่า “array”
  - ชื่อตัวแปร
  - เครื่องหมาย “#”
  - ตำแหน่งของตัวแปร หรือ ขนาดของตัวแปร

โดยแต่ละส่วนจะมีโทเค็น “UNDER” (_) เป็นตัวแบ่งส่วนเช่น main_integer_array_xx#3

“OPERATOR” คือโปรดักชันที่มีการคำนวณกันหรือส่งค่าออกมา สามารถแบ่งได้เป็น 2 กรณีคือ

กรณีมี Operand 1 ตัว (“ATOM”) คือกรณีที่ไม่มีการคำนวณและจะดึงค่าออกมาจาก operand นี้โดยตรง

กรณีมี Operand 2 ตัว และ มี Operator 1 ตัว คือกรณีที่มีการคำนวณเกิดขึ้น โดยจะแบ่งการคำนวณออกเป็น 3 แบบคือ

- คำนวณตัวเลข (+, -, *, /)
- คำนวณตรรกะ (and, or, greater, lesser, xor, etc.)
- คำนวณค่า bitstring (&, |, ~, etc.)

ภาคผนวก ค
รูปแบบภาษาสำหรับใช้แสดงภาพมโนทัศน์

รูปแบบภาษาสำหรับใช้แสดงภาพมโนทัศน์ มีรายละเอียดดังต่อไปนี้

1. ตัวแปร คือ ตำแหน่งหนึ่งในหน่วยความจำที่ใช้เก็บข้อมูล ตัวแปรจะต้องมีชนิดข้อมูลหนึ่ง (Type) กำกับไว้เสมอ ชนิดข้อมูลในภาษาสำหรับจินตทัศน์การคำนวณแบบเชื่อมตรงจะมีเพียง 3 ชนิด คือ ชนิดจำนวนเต็ม, ชนิดตรรกะ, ชนิดสายอักขระบิต

การตั้งชื่อตัวแปร มีหลักการตั้งชื่อสำหรับตัวแปร และฟังก์ชัน ดังนี้

- จะต้องขึ้นต้นด้วยตัวอักษร a ถึง z หรือ A ถึง Z จากนั้นตามตัวอักษรหรือตัวเลข 0 ถึง 9 และไวต่ออักษรใหญ่เล็ก (Case Sensitive) ห้ามมีการเว้นวรรค และต้องตั้งชื่อไม่ซ้ำกับคำสงวนหรือคีย์เวิร์ด ซึ่งแสดงดังตารางที่?
- ไม่จำกัดความยาว
- ห้ามมีการเว้นวรรค ห้ามตั้งชื่อซ้ำกัน

2. ชนิดข้อมูลและตัวดำเนินการ (Data Types and Operators)

ชนิดข้อมูล (Data Type) แบ่งออกเป็น 3 ชนิดดังนี้

- ชนิดจำนวนเต็ม (Integral type) ชนิดข้อมูลกลุ่มนี้จะใช้เก็บข้อมูลจำนวนเต็ม เช่น 123, 9239 เป็นต้น ค่าที่เก็บได้อยู่ระหว่าง -2,147,483,648 ถึง 2,147,483,647
- ชนิดตรรกะ (Logical type) ชนิดข้อมูลกลุ่มนี้จะใช้เก็บข้อมูลบูล (Boolean) เพื่อใช้เก็บค่าจริง (True) หรือเท็จ (False)
- ชนิดสายบิต (Bit string type) ชนิดข้อมูลกลุ่มนี้จะใช้เก็บข้อมูลสายบิต ค่าที่เก็บข้อมูลได้คือ 0 หรือ 1 เท่านั้นภายในเครื่องหมาย “ ยกตัวอย่างเช่น “101010” การเข้าถึงข้อมูลสามารถทำได้ 2 ลักษณะคือ การเข้าถึงทั้งชุดข้อมูลค่าที่ได้คือ “101010” หรือ การเข้าถึงโดยใช้ดรรชนีบอกตำแหน่งการเข้าถึง เช่น bitdata [1] ค่าข้อมูลที่เก็บอยู่คือ “1” เป็นต้น

3. ตัวดำเนินการและนิพจน์

สัญลักษณ์ที่ใช้ในการกระทำ (Operand) กับข้อมูลมีรูปแบบ หน้าที่และลักษณะการใช้งาน ดังตารางที่ ค-1

ตารางที่ ค-1 ตัวดำเนินการและนิพจน์

ตัวดำเนินการ	หน้าที่	รูปแบบการเรียกใช้งาน	ตัวอย่าง	ผลลัพธ์
=	กำหนดค่าให้กับตัวแปร	ตัวแปร = ค่าคงที่, การกระทำระหว่างสองนิพจน์	K = 199	K = 199
+	บวก	นิพจน์แรก + นิพจน์ที่สอง	5 + 3	8
-	ลบ	นิพจน์แรก – นิพจน์ที่สอง	12 - 4	8
*	คูณ	นิพจน์แรก * นิพจน์ที่สอง	4 * 2	8
/	หาร	นิพจน์แรก / นิพจน์ที่สอง	64 / 8	8
==	เปรียบเทียบว่าต้องเท่ากัน	นิพจน์แรก == นิพจน์ที่สอง	8 == 8	True
!=	เปรียบเทียบว่าต้องไม่เท่ากัน	นิพจน์แรก != นิพจน์ที่สอง	8 != 8	False
<	เปรียบเทียบว่าต้องน้อยกว่า	นิพจน์แรก < นิพจน์ที่สอง	9 > 8	True
<=	เปรียบเทียบว่าต้องน้อยกว่าเท่ากับ	นิพจน์แรก <= นิพจน์ที่สอง	8 <= 9	True
>	เปรียบเทียบว่าต้องมากกว่า	นิพจน์แรก > นิพจน์ที่สอง	8 > 9	False
>=	เปรียบเทียบว่าต้องมากกว่า	นิพจน์แรก >= นิพจน์ที่สอง	8 >= 9	False
not	กลับค่าจากจริงเป็นเท็จหรือค่าเท็จเป็นจริง	not นิพจน์	not (false)	true
and	เปรียบเทียบค่าบูล	นิพจน์แรก and นิพจน์ที่สอง	true and true	true
or	เปรียบเทียบค่าบูล	นิพจน์แรก or นิพจน์ที่สอง	true or false	true
xor	เปรียบเทียบค่าบูล	นิพจน์แรก xor นิพจน์ที่สอง	false xor true	true
~	ใช้กลับค่าอักขระบิต	~ นิพจน์	~”101”	“010”
&	เปรียบเทียบอักขระบิต	นิพจน์แรก & นิพจน์ที่สอง	“1” & “1”	“1”
|	เปรียบเทียบอักขระบิต	นิพจน์แรก | นิพจน์ที่สอง	“1” | “1”	“1”

ตารางที่ ค-1 (ต่อ) ตัวดำเนินการและนิพจน์

ตัว ดำเนินการ	หน้าที่	รูปแบบการเรียกใช้งาน	ตัวอย่าง	ผลลัพธ์
^	เปรียบเทียบอักขระบิต	นิพจน์แรก ^ นิพจน์ที่สอง	“0” ^ “1”	“1”
<<	เลื่อนบิตไปทางซ้าย	นิพจน์แรก << นิพจน์ที่สอง	“1010” << 3	01010000
>>	เลื่อนบิตไปทางขวา	นิพจน์แรก >> นิพจน์ที่สอง	“1010” >> 3	00000001
>>>	เลื่อนบิตไปทางขวาแต่ใส่ 0 จากซ้าย	นิพจน์แรก >>> นิพจน์ที่สอง	“1010” >>> 3	00000010

4. การประกาศใช้ฟังก์ชันมีสองรูปแบบ ดังนี้

- การประกาศฟังก์ชันแบบส่งค่ากลับจะต้องระบุชนิดของฟังก์ชันตามด้วยชื่อของฟังก์ชัน การส่งค่ากลับใช้คำสั่ง return โดยใช้ชนิดข้อมูลเดียวกับการประกาศชนิดฟังก์ชัน มีรูปแบบการใช้คำสั่ง return ต้องปรากฏอยู่ที่บรรทัดสุดท้ายของฟังก์ชันเสมอ เช่น

```
int sumvalue (int a,int b)
{
    int result;
    a = a +1;
    result a+ b;
}
```

- การประกาศฟังก์ชันแบบไม่ส่งค่ากลับ ใช้ void นำหน้าชื่อฟังก์ชัน เช่น void passvalue (int a)

```
{
  a = a +1;
}
```

โดยที่พารามิเตอร์ของฟังก์ชันทั้งหมดใช้รูปแบบการผ่านโดยค่า (Pass by value)

5. คำสั่งเงื่อนไข (Condition Statement) ใช้สำหรับการตัดสินใจที่จะเลือกทำหรือไม่ทำคำสั่งชุดใดชุดหนึ่ง ได้แก่ คำสั่ง if , คำสั่ง if-else มีรูปแบบคำสั่งดังนี้

- คำสั่ง if

  ใช้ตัดสินใจว่าจะทำหรือไม่ทำชุดคำสั่งหนึ่ง

```
If (condition – a Boolean expression)
  {
    ประโยคคำสั่ง…
  }
```

- คำสั่ง if - else

```
If (condition – a Boolean expression)
    {
      ประโยคคำสั่ง…
    }
  Else
  {
    ประโยคคำสั่ง…
  }
```

6. คำสั่งวนซ้ำ (Loop Statement) คำสั่งวนซ้ำในภาษา จะใช้วนซ้ำหรือไม่ ขึ้นอยู่กับผลลัพธ์ของเงื่อนไขที่เป็นบูล โดยจะวนซ้ำถ้าเงื่อนไขเป็นจริง มีรูปแบบคำสั่งดังนี้

```
While (condition – Boolean expression)
    {
      ประโยคคำสั่ง…
    }
```

- แถวลำดับ (Array) คือ โครงสร้างข้อมูลที่ใช้เก็บข้อมูลชนิดเดียวกัน ซึ่งในงานวิจัยนี้สามารถใช้แถวลำดับได้แบบขนาด 1 มิติ และเป็นแถวลำดับแบบคงที่มีรูปแบบคำสั่งดังนี้

Integer a[5];

Boolean b[2];

a [1] = 1;

a [2] = 2;

b [1] = ‘true’;

b [1] = ‘false’;

- ขอบเขตการประกาศและเรียกใช้ตัวแปรแบบท้องถิ่นอยู่ภายใต้การประกาศฟังก์ชัน
- Semicolon ใช้เป็นเครื่องหมายคั่นประโยค (Statement Separators)
- ฟังก์ชันภายใน (Built-in Functions) แบ่งออกเป็น 5 ฟังก์ชันดังนี้

-digitget คือ ฟังก์ชันการเรียกใช้ค่าจากตัวแปรนำเข้าโดยจะทำการดึงค่าจากตัวแปรนำเข้าทีละหลักนำมาประมวลผลผ่านอัลกอริทึม มีรูปแบบคำสั่งดังนี้

```
digitget [parameter];
```

โดยที่ parameter คือตัวแปรนำเข้า ซึ่งมีชนิดข้อมูลเป็นชนิดจำนวนเต็มที่มีช่วงข้อมูลระหว่าง -1,0,1

ตัวอย่าง เช่น xx = digitget x;

-bitstrtoint คือ ฟังก์ชันสำหรับแปลงค่าสายบิตอักขระเป็นข้อมูลชนิดจำนวนเต็ม มีรูปแบบคำสั่งดังนี้

```
bitstrtoint(parameter);
```

โดยที่ parameter คือสายบิตอักขระ เช่น “110”,”111” เป็นต้น

ตัวอย่าง เช่น bitstrtoint (“110”) = 5

-inttobitstr คือ ฟังก์ชันสำหรับแปลงค่าจำนวนเต็มไปเป็นข้อมูลชนิดสายอักขระบิต มีรูปแบบคำสั่งดังนี้

```
inttobitstr (parameter1, parameter2);
```

โดยที่ parameter1 คือตัวเลขจำนวนเต็ม เช่น 10, 99 เป็นต้น

parameter2 คือตัวเลขจำนวนหลักของสายบิตอักขระ เช่น 5

ตัวอย่าง เช่น inttobitstr(5,4) = “0110”

-inttosbitstr คือ ฟังก์ชันสำหรับแปลงค่าจำนวนเต็มแบบมีเครื่องหมายไปเป็นข้อมูลชนิดสายอักขระบิต มีรูปแบบคำสั่งดังนี้

```
inttosbitstr (parameter1, parameter2);
```

โดยที่ parameter1 คือตัวเลขจำนวนเต็มแบบมีเครื่องหมาย เช่น 5, -5 เป็นต้น

parameter2 คือตัวเลขจำนวนหลักของสายบิตอักขระ เช่น 6

ตัวอย่าง เช่น inttosbitstr (-5,6) = “010001”

-sbitstroint คือ ฟังก์ชันสำหรับแปลงค่าข้อมูลชนิดสายอักขระบิตไปเป็นจำนวนเต็มแบบมีเครื่องหมาย มีรูปแบบคำสั่งดังนี้

```
sbitstroint (parameter);
```

โดยที่ parameter คือ สายบิตอักขระ เช่น 010001 เป็นต้น

ตัวอย่าง เช่น sbitstroint (‘010001’) = -5

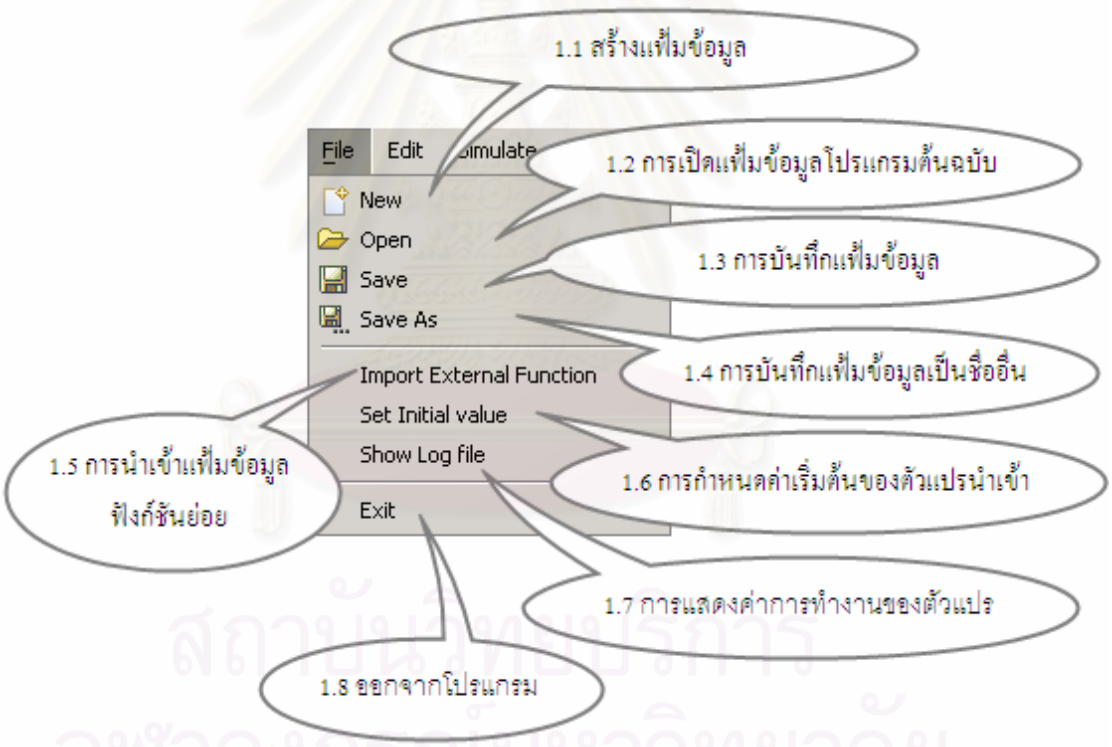
คำอธิบาย (Comments) ใช้เครื่องหมาย /* …. ข้อความ .... */ เป็นวิธีการละข้อความสำหรับข้อความขนาดยาว และเครื่องหมาย // เพื่อละข้อความที่มีความยาวไม่เกิน 1 บรรทัด

ภาคผนวก ง
วิธีการใช้เครื่องมือ

การใช้งานเครื่องมือสร้างภาพมโนทัศน์

การใช้งานเครื่องมือแสดงภาพมโนทัศน์สำหรับช่วยออกแบบการคำนวณแบบเชื่อมตรงระดับดิจิตสำหรับผู้ออกแบบอัลกอริทึม แบ่งออกเป็น 6 ส่วนคือ การจัดการกับแฟ้มข้อมูล การแก้ไขแฟ้มข้อมูลโปรแกรมต้นฉบับ การจัดการกับตัวแปรนำเข้าและผลลัพธ์ การเขียนโปรแกรมต้นฉบับ การควบคุมการแสดงภาพมโนทัศน์ ดังนี้

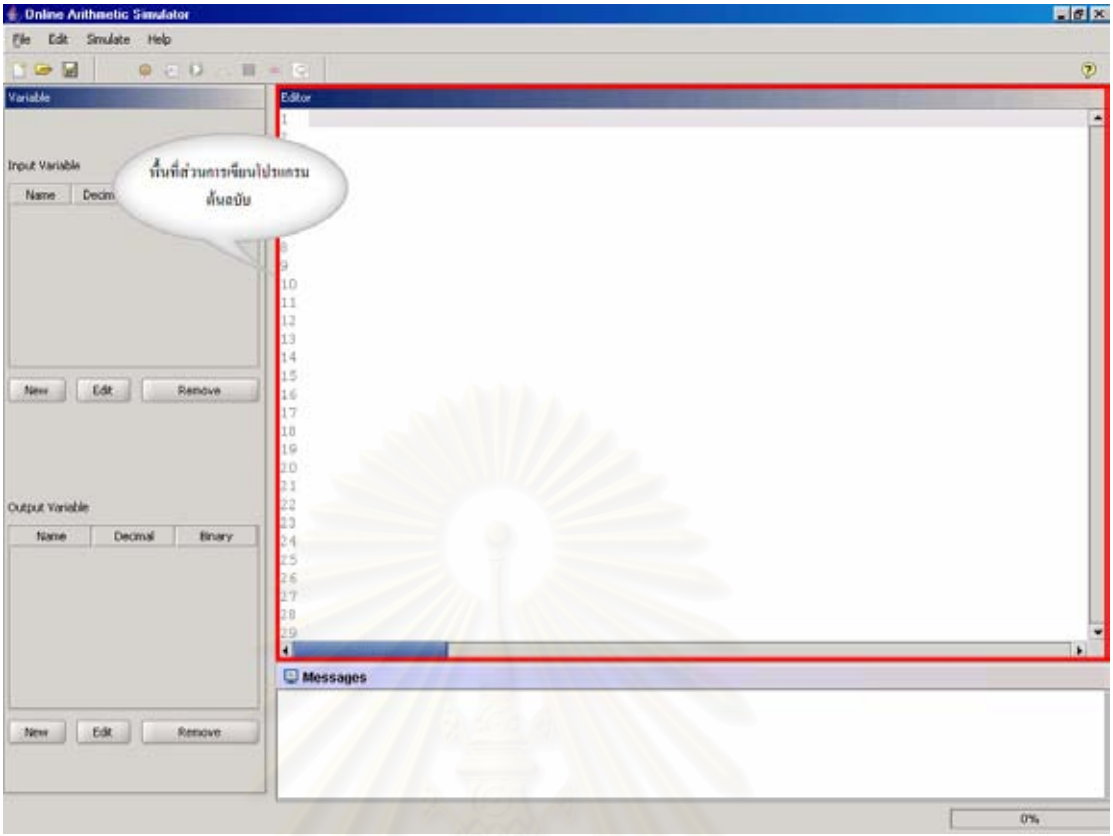
1. การจัดการกับแฟ้มข้อมูล การจัดการกับแฟ้มข้อมูลแบ่งออกเป็น 6 ส่วนคือ การสร้างแฟ้มข้อมูลโปรแกรมต้นฉบับใหม่ การเปิดแฟ้มข้อมูลโปรแกรมต้นฉบับ การบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับ การบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับเป็นชื่ออื่น การนำเข้าแฟ้มฟังก์ชันย่อยจากภายนอก การกำหนดค่าเริ่มต้นของตัวแปรนำเข้า การแสดงค่าการทำงานของตัวแปร การออกจากโปรแกรม ดังรูปที่ ง-1



รูปที่ ง-1 เมนูการจัดการกับแฟ้มข้อมูลโปรแกรมต้นฉบับ

1.1 การสร้างแฟ้มข้อมูลใหม่ (New) มีขั้นตอนดังนี้

- เลือกไปที่เมนูหลัก “File” และคลิกที่เมนู “New”ดังรูปที่ ง-1
- เครื่องมือจะแสดงพื้นที่ส่วนการเขียนโปรแกรมต้นฉบับ ดังรูปที่ ง-2

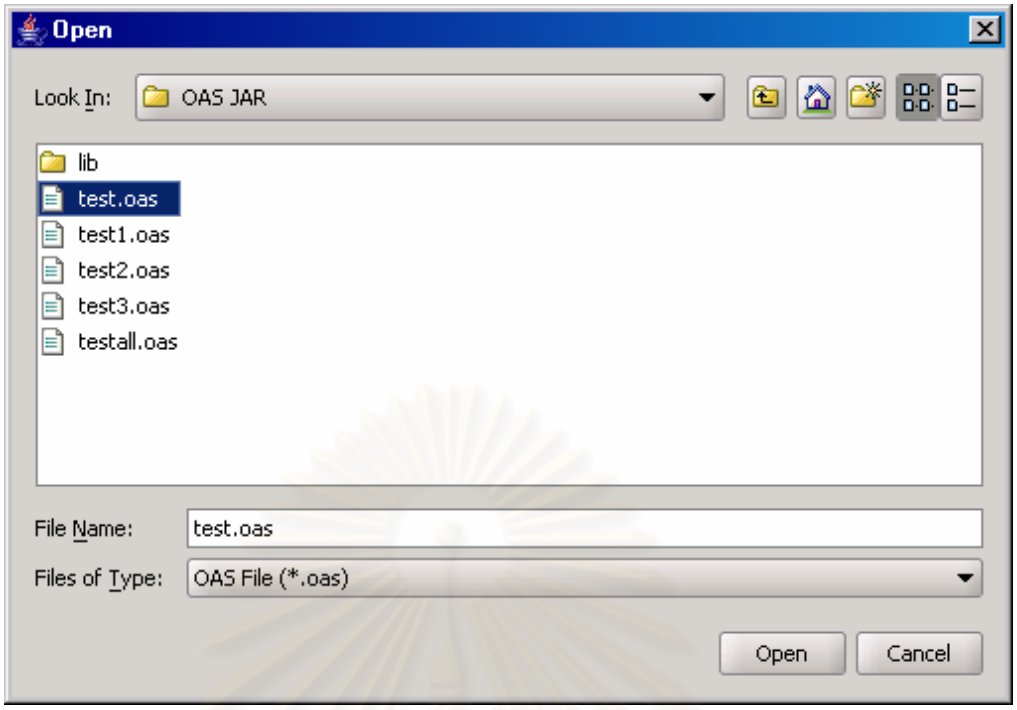


รูปที่ ง-2 พื้นที่ส่วนการเขียนโปรแกรมต้นฉบับ

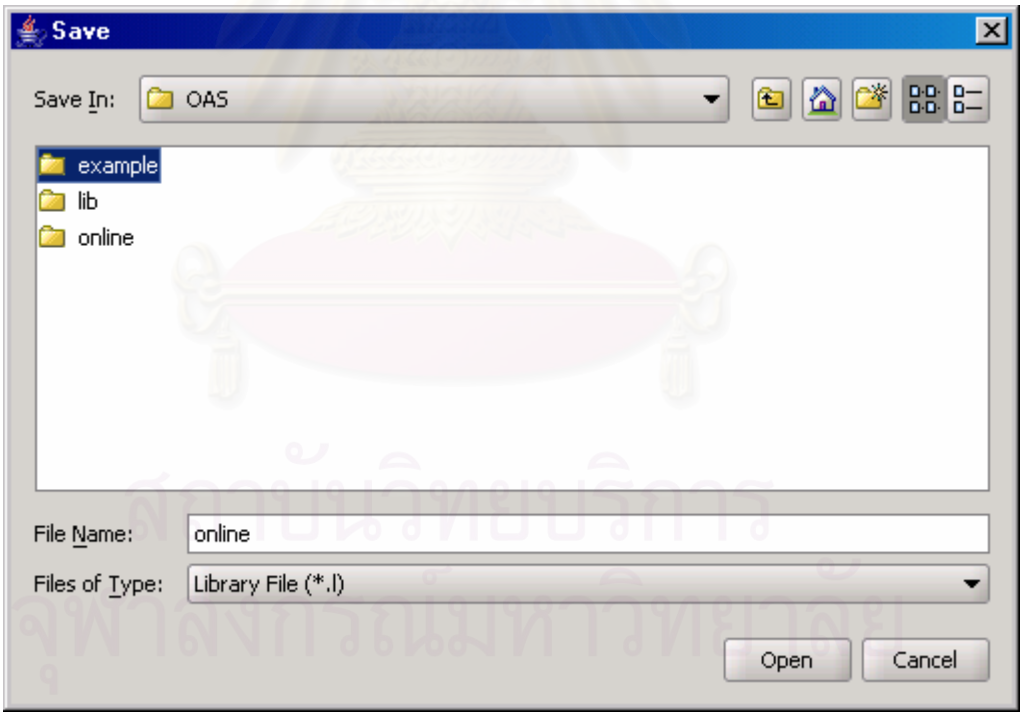
1.2 การเปิดแฟ้มข้อมูลโปรแกรมต้นฉบับเดิม (Open) มีขั้นตอนดังนี้

- เลือกที่เมนูหลัก “File” และคลิกเมนู “Open” ดังรูปที่ ง-1
- จากนั้นเครื่องมือจะแสดงหน้าจอให้เลือกรายการแฟ้มข้อมูลที่มีในระบบ

โดยคลิกเลือกแฟ้มข้อมูล และคลิกปุ่ม Open ดังรูปที่ ง-3



รูปที่ ง-3 หน้าจอเลือกรายการแฟ้มโปรแกรมต้นฉบับ



รูปที่ ง-4 หน้าจอบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับ

1.3 การบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับ (Save) มีขั้นตอนดังนี้

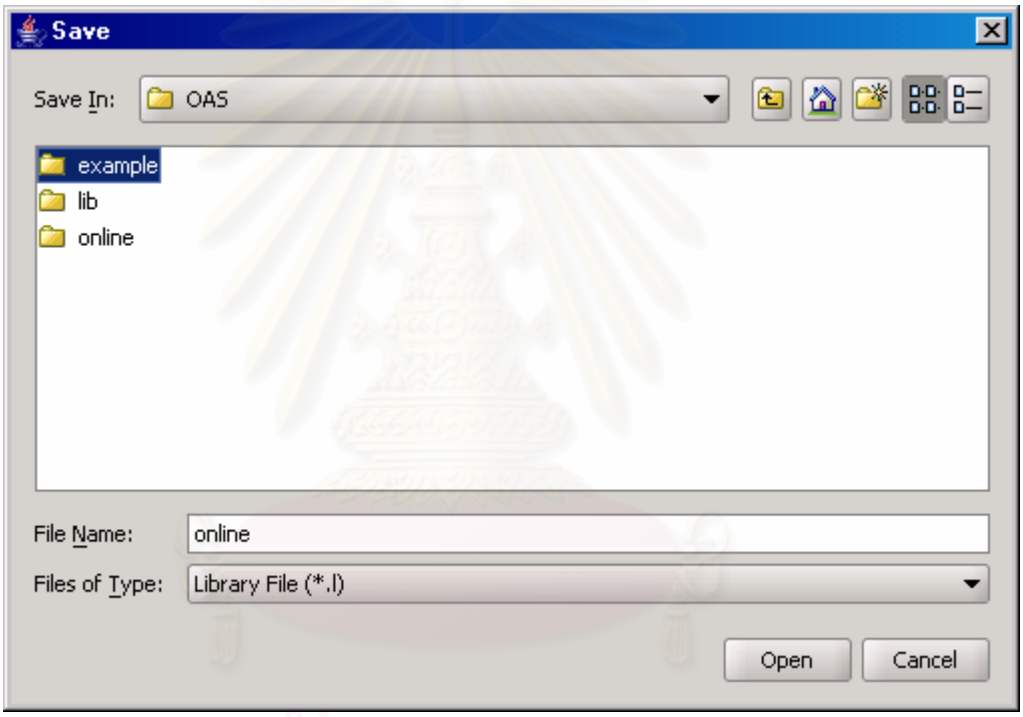
- เลือกไปที่เมนูหลัก “File” และคลิกเมนู “Save” ดังรูปที่ ง-1

- จากนั้นเครื่องมือจะแสดงหน้าจอบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับโดยคลิกที่ปุ่ม Save ดังรูปที่ ง-4

1.4 การบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับเป็นชื่ออื่น (Save As) มีขั้นตอนดังนี้

- เลือกไปที่เมนูหลัก “File” และคลิกเมนู “Save As” ดังรูปที่ ง-1

- จากนั้นเครื่องมือจะแสดงหน้าจอบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับเป็นชื่ออื่นโดยคลิกปุ่ม Save As ดังรูปที่ ง-5

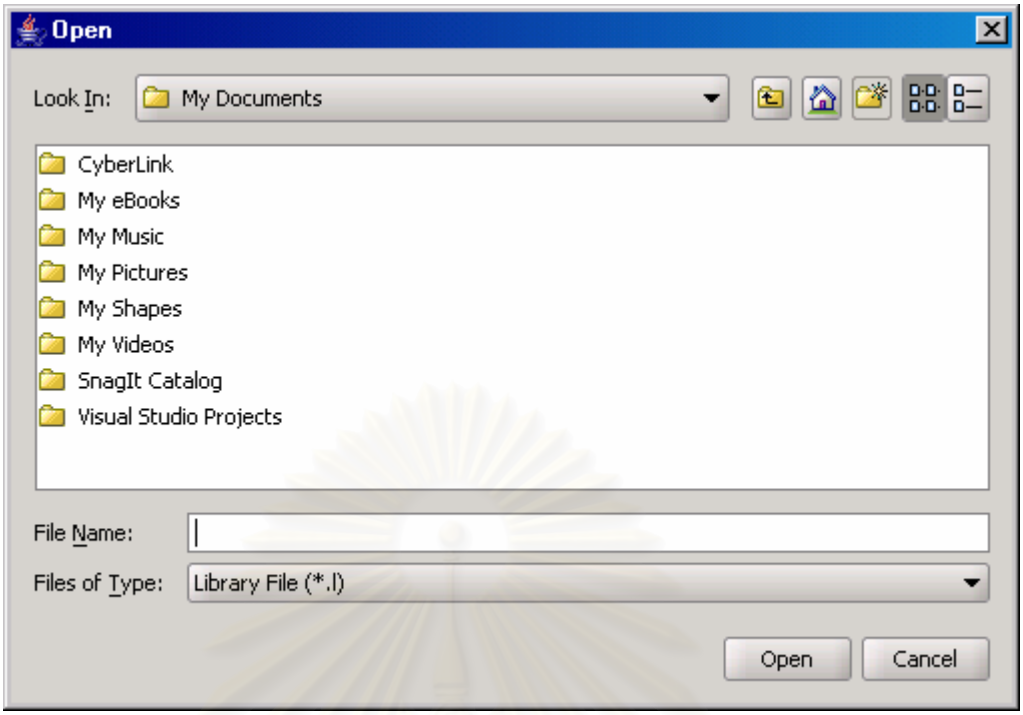


รูปที่ ง-5 หน้าจอบันทึกแฟ้มข้อมูลโปรแกรมต้นฉบับเป็นชื่ออื่น

1.5 การนำเข้าแฟ้มฟังก์ชันย่อยจากภายนอก (Import External Function) มีขั้นตอนดังนี้

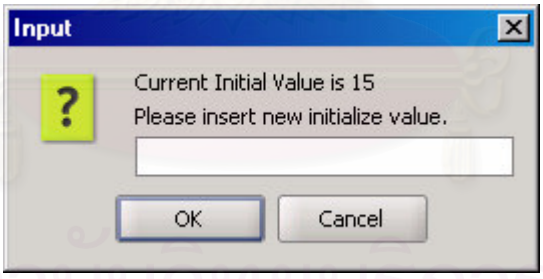
- เลือกไปที่เมนูหลัก “File” และคลิกเมนู “Import External Function” ดังรูปที่ ง-1

- จากนั้นเครื่องมือจะแสดงหน้าจอแสดงรายการแฟ้มข้อมูลฟังก์ชันย่อยที่มีในระบบ เลือกชื่อแฟ้มข้อมูลแฟ้มข้อมูลฟังก์ชันย่อยที่ต้องการเปิดโดยคลิกที่ “Select” และคลิกปุ่ม Open ดังรูปที่ ง-6



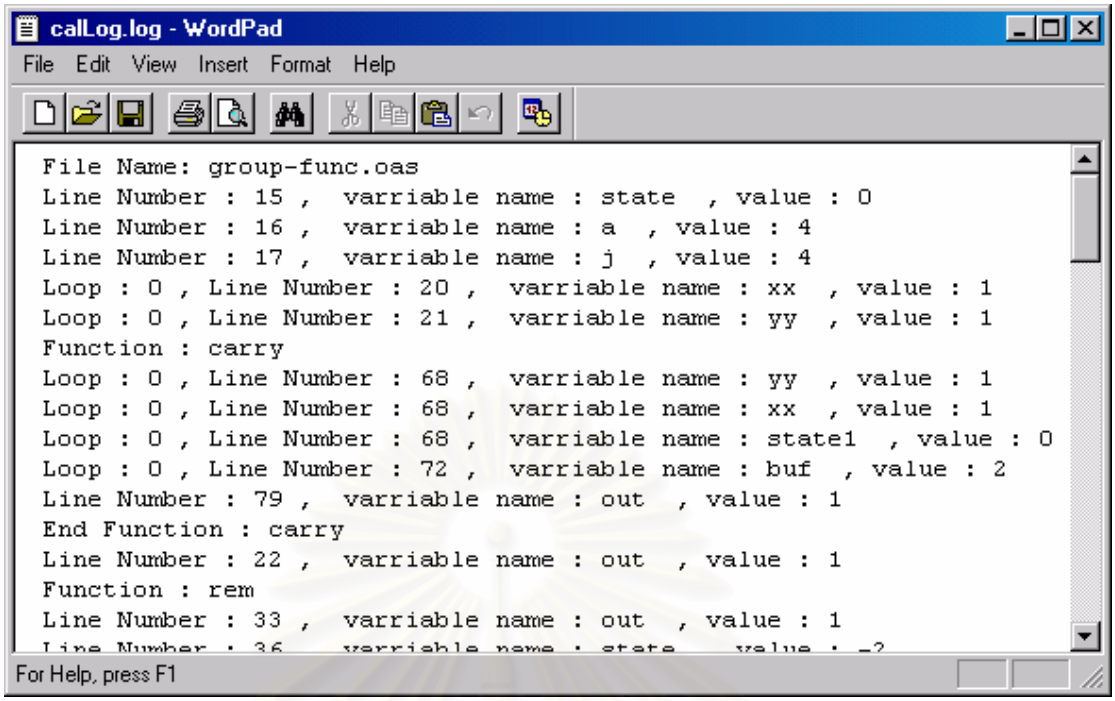
รูปที่ ง-6 หน้าจอเลือกแฟ้มข้อมูลฟังก์ชันย่อย

1.6 การกำหนดค่าเริ่มต้นของตัวแปรนำเข้า (Set Initial value) เป็นการกำหนดค่าเริ่มต้นของตัวแปรนำเข้าโดยสามารถปรับเปลี่ยนได้จากผู้ออกแบบ ทำได้โดยเลือกไปที่เมนูหลัก “File” และคลิกเมนู “Set Initial value” ดังรูปที่ ง-7



รูปที่ ง-7 หน้าจอกำหนดค่าเริ่มต้นของตัวแปรนำเข้า

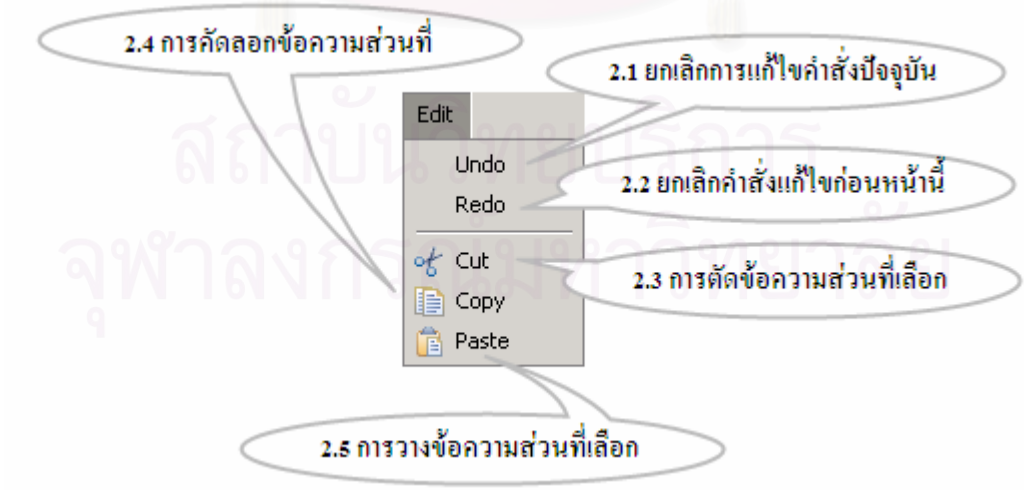
1.7 การแสดงค่าการทำงานของตัวแปร (Show Log File) เป็นการตรวจสอบการทำงานของตัวแปร ทำได้โดยเลือกไปที่เมนูหลัก “File” และคลิกเมนู “(Show Log File” ดังรูปที่ ง-8



รูปที่ ง-8 หน้าจอแสดงรายละเอียดค่าของตัวแปรจากการแสดงการทำงาน

1.6 การออกจากระบบ (Exit) เป็นการออกจากระบบเพื่อเลิกการใช้งานเครื่องมือ ทำได้โดยเลือกไปที่เมนูหลัก “File” และคลิกเมนู “Exit” ดังรูปที่ ง-1

2. การแก้ไขแฟ้มข้อมูลโปรแกรมต้นฉบับ การจัดการกับแฟ้มข้อมูลแบ่งออกเป็น 5 ส่วนคือ ยกเลิกการแก้ไขคำสั่งปัจจุบัน การยกเลิกคำสั่งแก้ไขก่อนหน้านี้ การตัดข้อความส่วนที่เลือก การคัดลอกข้อความส่วนที่เลือก การวางข้อความส่วนที่เลือก ดังรูปที่ ง-9



รูปที่ ง-9 เมนูแก้ไขแฟ้มข้อมูลโปรแกรมต้นฉบับ

2.1 ยกเลิกการแก้ไขคำสั่งปัจจุบัน (Undo)

- เลือกไปที่เมนูหลัก “Edit” และคลิกเมนู “Undo” ดังรูปที่ ง-9

2.2 การยกเลิกคำสั่งแก้ไขก่อนหน้านี้ (Redo)

- เลือกไปที่เมนูหลัก “Edit” และคลิกเมนู “Redo” ดังรูปที่ ง-9

2.3 การตัดข้อความส่วนที่เลือกแถบสี (Cut)

- เลือกไปที่เมนูหลัก “Edit” และคลิกเมนู “Cut” ดังรูปที่ ง-9

2.4 การคัดลอกข้อความส่วนที่เลือกแถบสี

- เลือกไปที่เมนูหลัก “Edit” และคลิกเมนู “Copy” ดังรูปที่ ง-9

2.5 การวางข้อความส่วนที่เลือก

- เลือกไปที่เมนูหลัก “Edit” และคลิกเมนู “Paste” ดังรูปที่ ง-9

3. การจัดการกับตัวแปรนำเข้าและตัวแปรผลลัพธ์ ทำงานผ่านทางส่วนรับตัวแปรนำเข้าและตัวแปรผลลัพธ์ด้านซ้ายมือของส่วนการเขียนโปรแกรม ดังรูปที่ ง-10 การทำงานแบ่งเป็น 2 ส่วนดังนี้

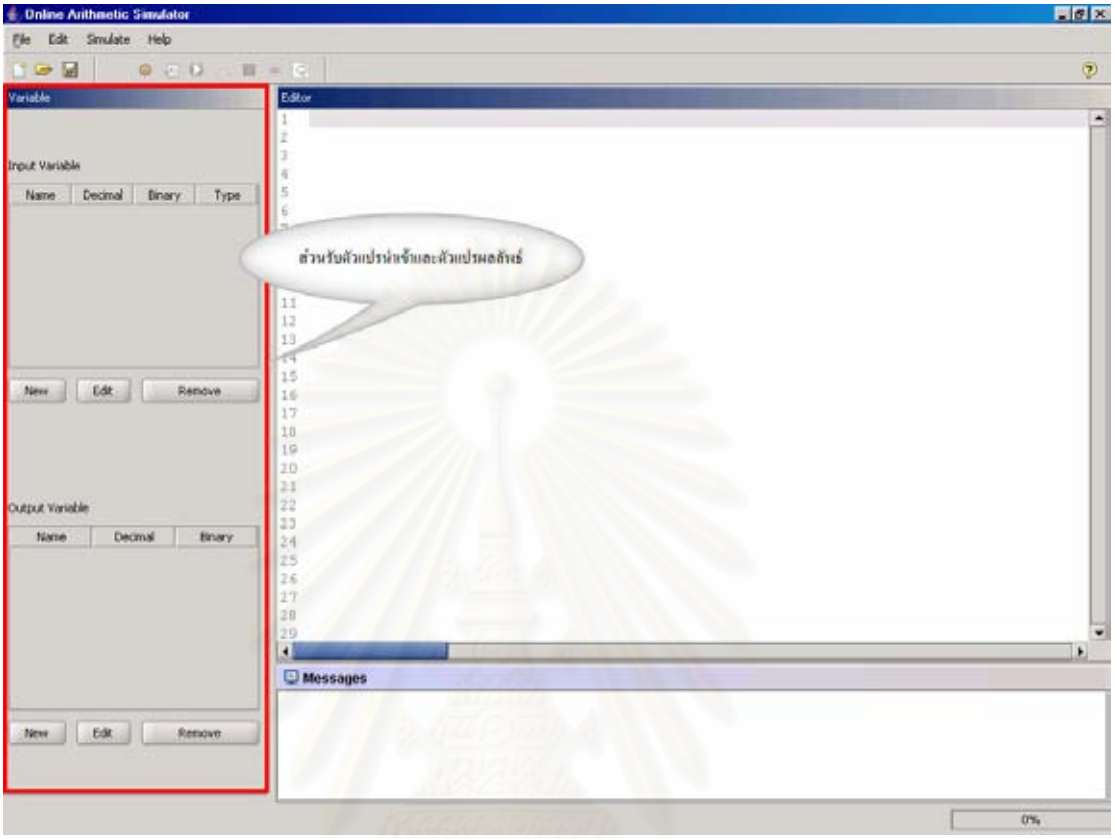
3.1 การป้อนตัวแปรนำเข้า (Input Variables) แบ่งการทำงานเป็น 3 ส่วนคือ การป้อนตัวแปรนำเข้า การแก้ไขตัวแปรนำเข้า การลบตัวแปรนำเข้า

3.1.1 การป้อนตัวแปรนำเข้า มีขั้นตอนดังนี้

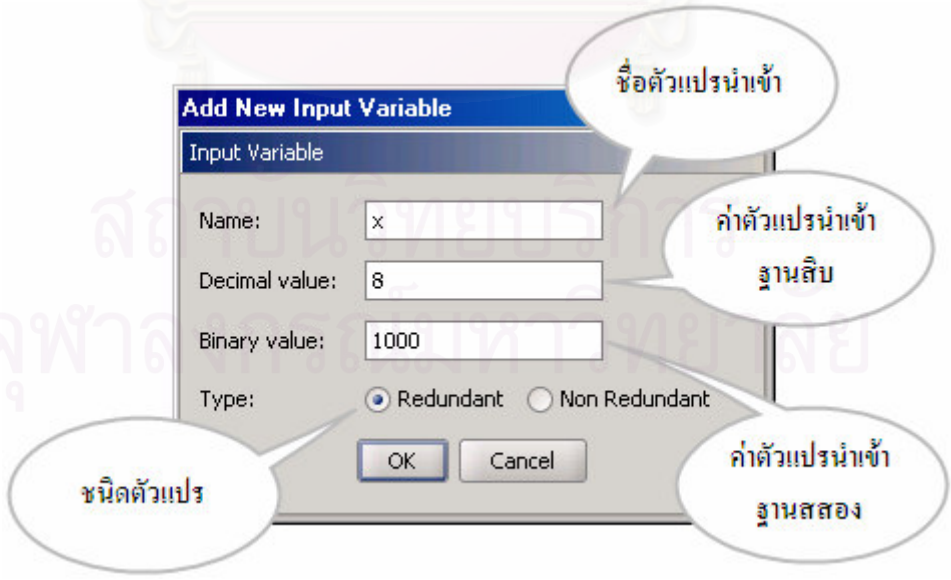
1) คลิกที่ปุ่ม “New” เครื่องมือจะแสดงกรอบรับตัวแปรนำเข้า ประกอบด้วย คือ ชื่อตัวแปรนำเข้า, ค่าตัวแปรนำเข้ารูปแบบจำนวนเต็มเลขฐานสิบ (Decimal Value), ค่าตัวแปรนำเข้ารูปแบบเลขฐานสอง (Binary Value), ชนิดของตัวแปร แบ่งเป็น 2 ประเภท คือ Redundant และ Non-Redundant การทำงานของเครื่องมืออยู่บนพื้นฐานของดิจิตบนเลขฐานสอง ดังรูปที่ ง-11

กรณีเลือก Redundant เพื่อกำหนดข้อมูลทำให้สามารถเลือกข้อมูลดิจิตในแบบเลขฐานสองแบบมีเครื่องหมายมีค่าข้อมูลที่เป็นไปได้คือ -1, 0, 1

กรณีเลือก Non-Redundant เพื่อกำหนดข้อมูลทำให้สามารถเลือกข้อมูลดิจิตรูปแบบเลขฐานสอง มีค่าข้อมูลที่เป็นไปได้คือ 0, 1



รูปที่ ง-10 ส่วนรับตัวแปรนำเข้าและตัวแปรผลลัพธ์



รูปที่ ง-11 การป้อนค่าตัวแปรนำเข้า

เมื่อกำหนดข้อมูลตัวแปรนำเข้าเสร็จเรียบร้อยแล้วกดปุ่ม “OK” ยืนยันการป้อนข้อมูล

3.1.2 การแก้ไขตัวแปรนำเข้า

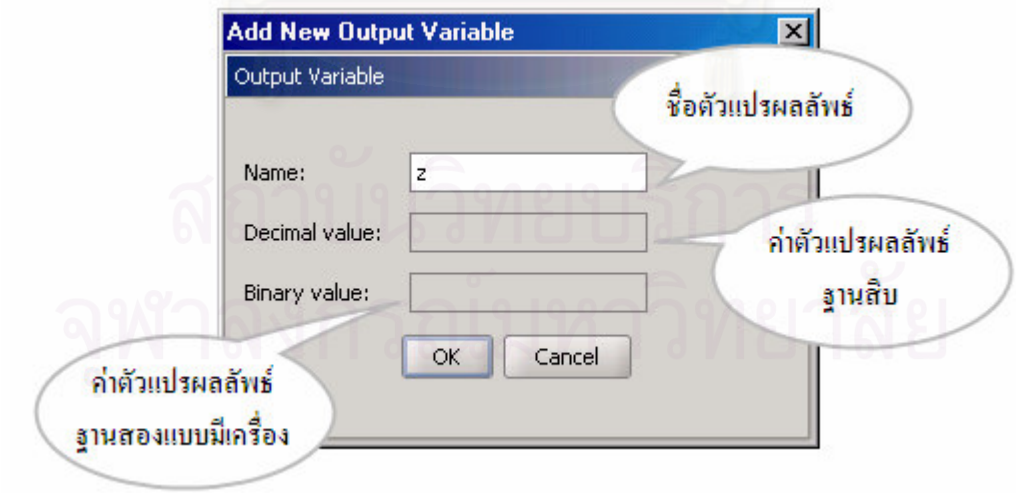
a) คลิกที่ปุ่ม “Edit” เครื่องมือจะแสดงกรอบสำหรับแก้ไขตัวแปรเมื่อแก้ไขข้อมูลที่ต้องการเสร็จเรียบร้อยแล้ว กดปุ่ม “UPDATE” ยืนยันการปรับปรุงข้อมูลตัวแปรนำเข้า ดังรูปที่ ง-10

3.1.3 การลบตัวแปรนำเข้า คลิกเลือกแถบสีที่ตัวแปรนำเข้า กดปุ่ม “Remove” ตัวแปรที่ต้องการลบ ดังรูปที่ ง-10

3.2 การป้อนตัวแปรผลลัพธ์ (Output Variables) แบ่งการทำงานเป็น 3 ส่วนคือ การป้อนตัวแปรผลลัพธ์ การแก้ไขตัวแปรผลลัพธ์ การลบตัวแปรนำผลลัพธ์ ดังรูปที่ ง-10

3.2.1 คลิกที่ปุ่ม “New” เครื่องมือจะแสดงกรอบรับตัวแปร ประกอบด้วยส่วนรับข้อมูล 3 ส่วน คือ ส่วนรับชื่อตัวแปรนำเข้า (Name), ส่วนแสดงผลลัพธ์เลขจำนวนเต็มฐานสิบ (Decimal Value), ส่วนแสดงผลลัพธ์รูปแบบเลขฐานสอง (Binary Value)

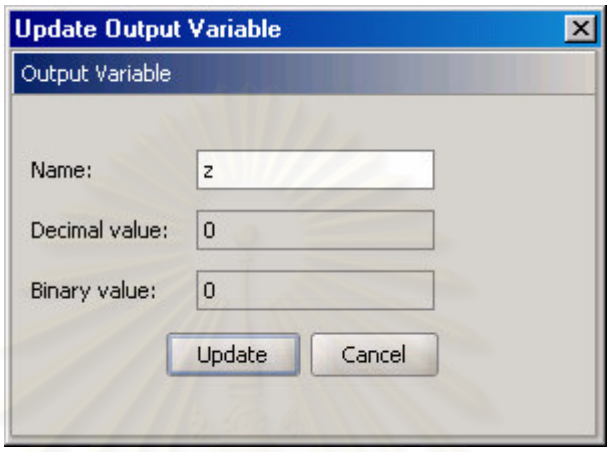
เมื่อกำหนดชื่อข้อมูลในส่วนของกรอบรับข้อมูลผลลัพธ์เสร็จเรียบร้อยแล้วกดปุ่ม “OK” ยืนยันการป้อนข้อมูลตัวแปรผลลัพธ์ ดังรูปที่ ง-12



รูปที่ ง-12 หน้าต่างรับตัวแปรผลลัพธ์

3.2.2 การแก้ไขตัวแปรผลลัพธ์

a) คลิกที่ปุ่ม “Edit” เครื่องมือจะแสดงกรอบสำหรับแก้ไขตัวแปรเมื่อแก้ไขข้อมูลที่ต้องการเสร็จเรียบร้อยแล้ว กดปุ่ม “UPDATE” ยืนยันการปรับปรุงข้อมูลตัวแปรผลลัพธ์ ดังรูปที่ ง-13

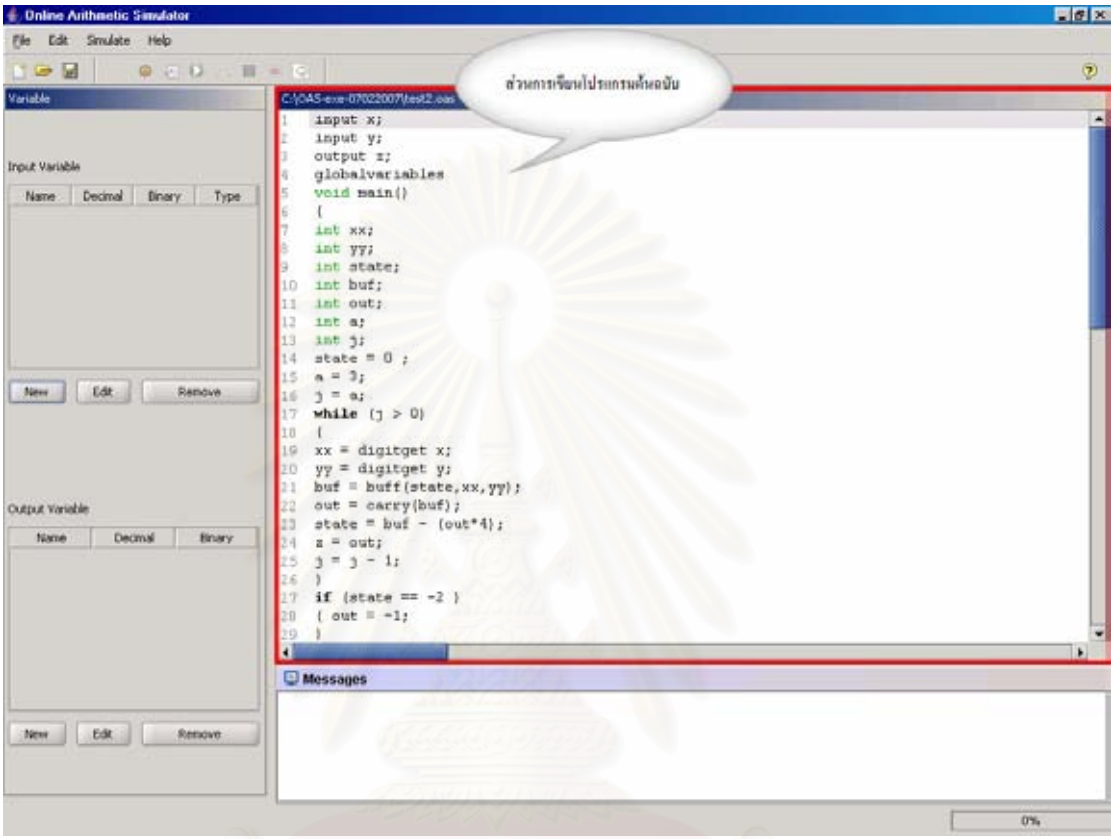


รูปที่ ง-13 หน้าต่างแก้ไขตัวแปรผลลัพธ์

3.2.3 การลบตัวแปรผลลัพธ์ คลิกเลือกตัวแปรผลลัพธ์ กดปุ่ม “Remove” ตัวแปรที่ต้องการลบ ดังรูปที่ ง-10

4. การเขียนโปรแกรมต้นฉบับ

เป็นส่วนสำหรับผู้ออกแบบอัลกอริทึมใช้ในการเขียนหรือแก้ไขโปรแกรมต้นฉบับก่อนสั่งคอมไพล์โปรแกรมเพื่อตรวจสอบความถูกต้อง แสดงดังรูปที่ ง-14

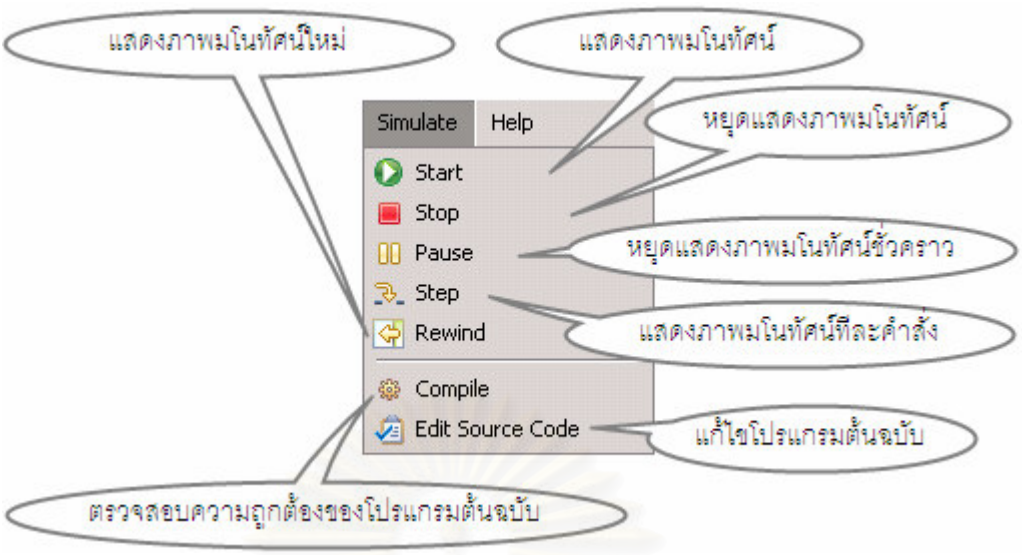


รูปที่ ง-14 ส่วนการเขียนโปรแกรมต้นฉบับ

5. การควบคุมการแสดงภาพมโนทัศน์ แบ่งการควมคุมการแสดงภาพมโนทัศน์ออกเป็น 7 ส่วนคือ แสดงภาพมโนทัศน์, หยุดการแสดงภาพมโนทัศน์, หยุดการแสดงภาพมโนทัศน์ชั่วคราว, แสดงภาพมโนทัศน์ครั้งละหนึ่งคำสั่ง, แสดงภาพมโนทัศน์ใหม่, ตรวจสอบความถูกต้องของโปรแกรมต้นฉบับ, แก้ไขโปรแกรมต้นฉบับ ดังรูปที่ ง-15

5.1 แสดงภาพมโนทัศน์ (Start) ใช้สำหรับสั่งให้แสดงภาพมโนทัศน์การทำงานสถานะการทำงานเปิดทาง (Enable) ถ้าพบว่าโปรแกรมต้นฉบับได้ผ่านการคอมไพล์ตรวจสอบความถูกต้องแล้ว มีขั้นตอนดังนี้

- เลือกไปที่เมนูหลัก “Simulate” และคลิกเมนู “Start” ดังรูปที่ ง-15



รูปที่ ง-15 เมนูควบคุมการแสดงภาพมโนทัศน์

5.2 หยุดแสดงภาพมโนทัศน์ (Stop) ใช้สำหรับสั่งหยุดการแสดงภาพมโนทัศน์ สถานะการทำงานเปิดทาง ถ้าพบว่าอยู่ในสถานะกำลังแสดงภาพมโนทัศน์ และกลับสู่สถานการณ์รอรับการสั่งแสดงภาพมโนทัศน์ใหม่ มีขั้นตอนดังนี้

- เลือกไปที่เมนูหลัก “Simulate” และคลิกเมนู “Stop” ดังรูปที่ ง-15

5.3 หยุดแสดงภาพมโนทัศน์ชั่วคราว (Pause) ใช้สำหรับสั่งหยุดการแสดงภาพมโนทัศน์ชั่วคราว โดยจะทำคำสั่งปัจจุบันจนจบคำสั่งและหยุดการทำงานเพื่อรอคำสั่งอื่นต่อไป มีขั้นตอนดังนี้

- เลือกไปที่เมนูหลัก “Simulate” และคลิกเมนู “Pause” ดังรูปที่ ง-15

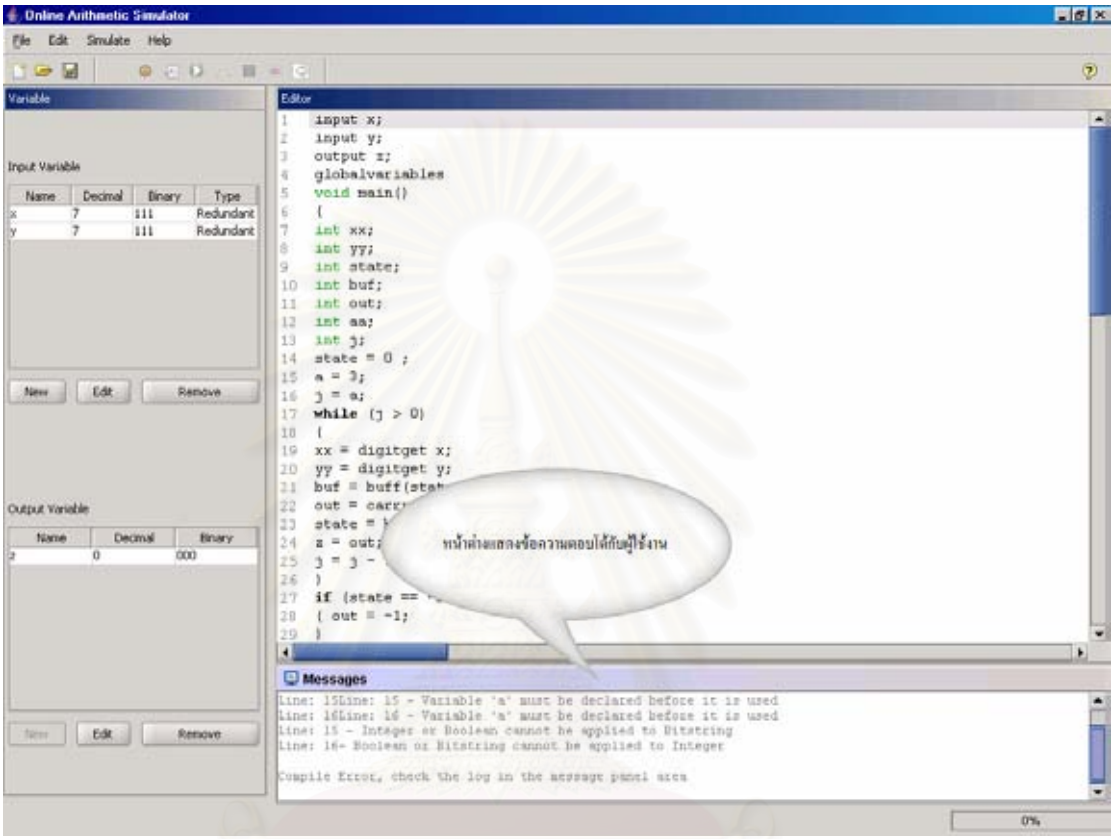
5.4 แสดงภาพมโนทัศน์ทีละคำสั่ง (Step) ใช้สำหรับสั่งแสดงภาพมโนทัศน์ทีละคำสั่ง เมื่อจบคำสั่งหนึ่งคำสั่งใด ก็จะรอการทำงานต่อไปจากผู้ใช้ มีขั้นตอนดังนี้

- เลือกไปที่เมนูหลัก “Simulate” และคลิกเมนู “Step” ดังรูปที่ ง-15

5.5 แสดงภาพมโนทัศน์ใหม่ (Rewind) ใช้สำหรับสั่งเริ่มแสดงภาพมโนทัศน์ใหม่ทั้งหมด มีขั้นตอนดังนี้

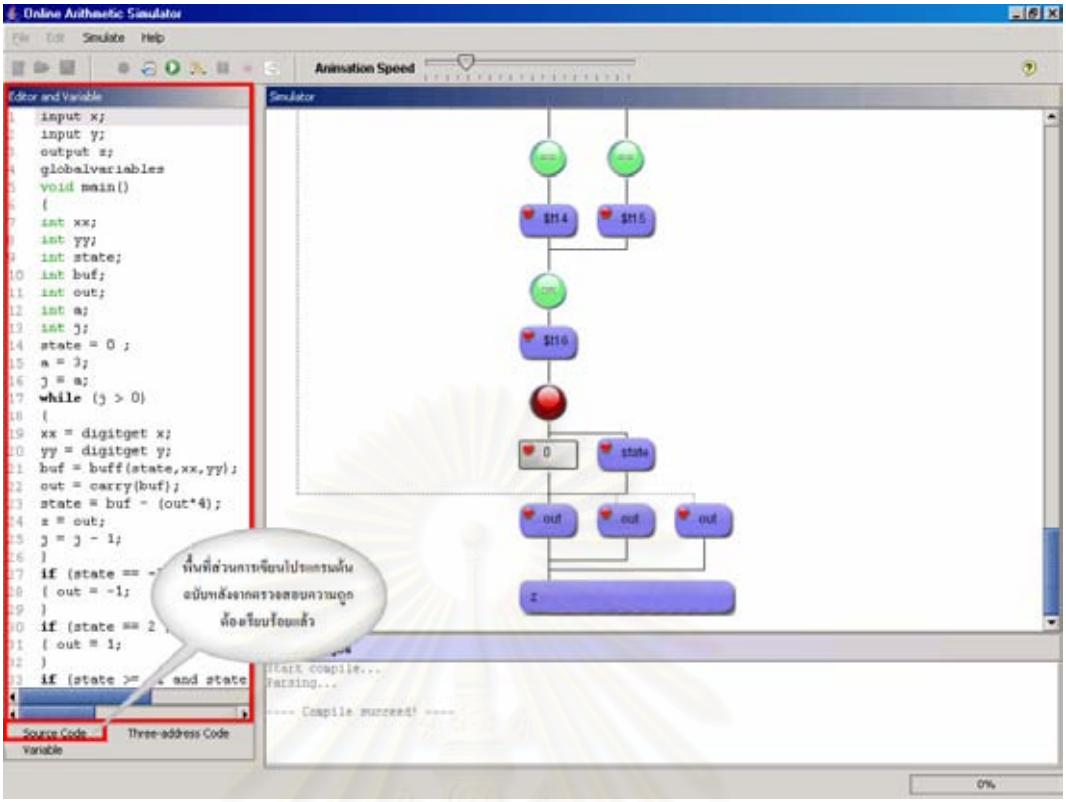
- เลือกไปที่เมนูหลัก “Simulate” และคลิกเมนู “Rewind” ดังรูปที่ ง-15

5.6 ตรวจสอบความถูกต้องของโปรแกรมต้นฉบับ (Compile) ใช้สำหรับตรวจสอบไวยากรณ์ภาษาว่าผู้ใช้ได้เขียนตามที่กำหนดไว้หรือไม่ โดยเลือกไปที่เมนูหลัก “Simulate” และคลิกเมนู “Compile” ดังรูปที่ ง-15 ถ้าพบว่าไม่ถูกต้องจะแสดงข้อความผิดพลาดให้เห็นในส่วนแสดงข้อความตอบโต้กับผู้ใช้ ดังรูปที่ ง-16

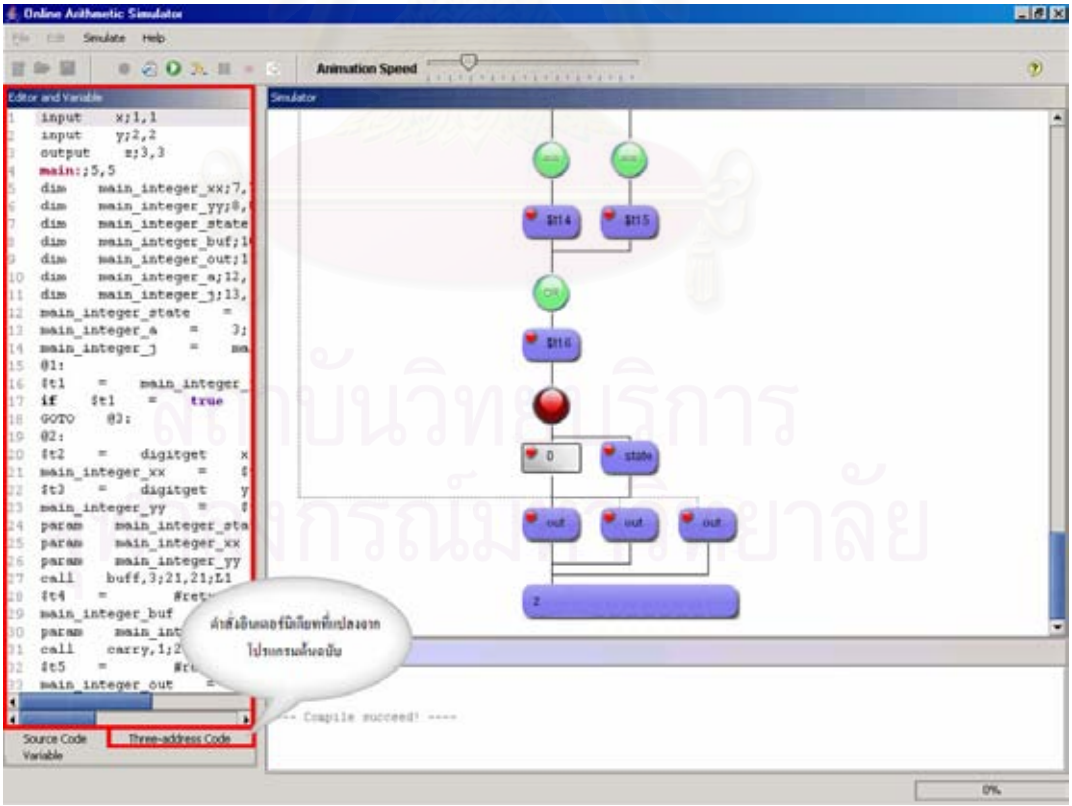


รูปที่ ง-16 หน้าต่างแสดงข้อความตอบโต้กับผู้ใช้

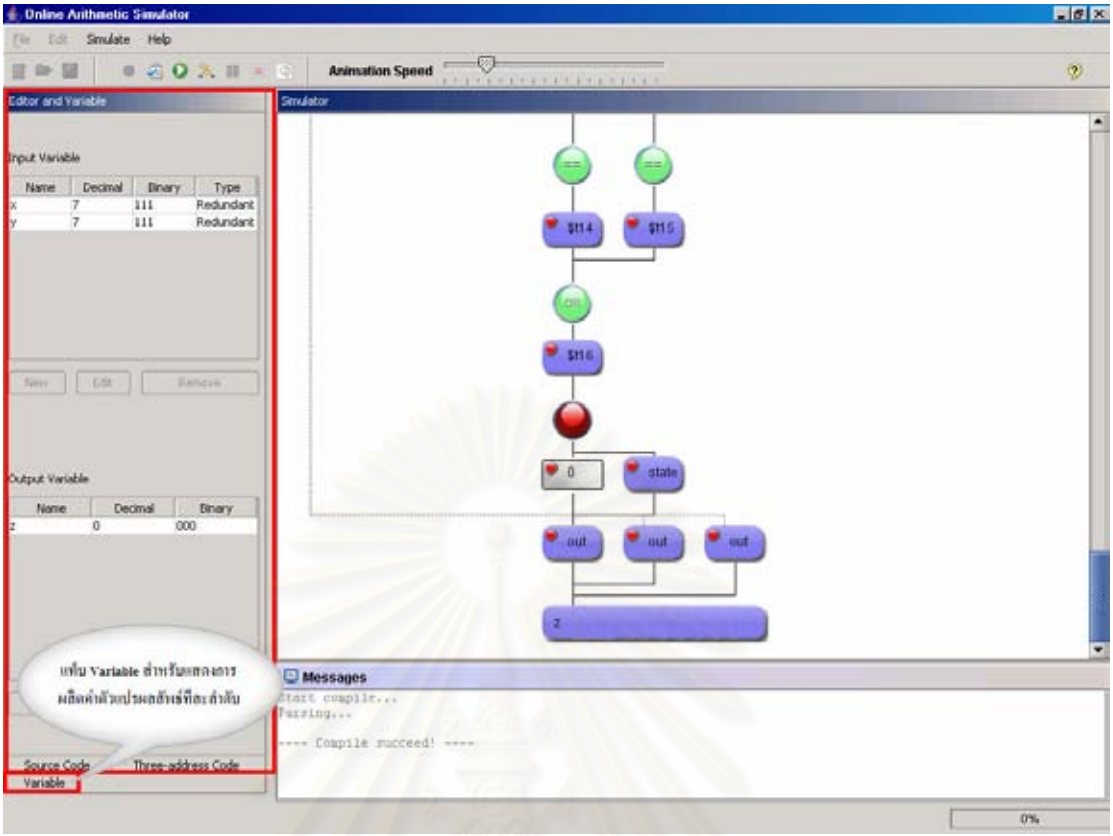
ถ้าพบว่าเครื่องมืออยู่ในสถานะตรวจสอบความถูกต้องของโปรแกรมต้นฉบับเรียบร้อยแล้วจากตัวแปลภาษา พื้นที่ส่วนการเขียนโปรแกรมต้นฉบับจะถูกย้ายไปอยู่ทางด้านซ้ายมือของหน้าจอหลัก ภายในแท็บ “Source Code” ดังรูปที่ ง-17 พร้อมกับส่วนแสดงคำสั่งอินเตอร์มิเดียทที่ได้รับจากการแปลงโปรแกรมต้นฉบับแสดงภายในแท็บ “Three-address Code” ดังรูปที่ ง-18 และแสดงแท็บ “Variable” แสดงค่าของตัวแปรนำเข้า และตัวแปรผลลัพธ์ โดยที่ค่าตัวแปรผลลัพธ์ จะแสดงคำตอบที่ถูกผลิตออกมาทีละลำดับจากการทำงานของอัลกอริทึม ดังรูปที่ ง-19



รูปที่ ง-17 แท็บ “Source Code”



รูปที่ ง-18 แท็บ “Three-address Code”



รูปที่ ง-19 แท็บ “Variable”

5.7 การแก้ไขโปรแกรมต้นฉบับ (Edit Source Code) ใช้สำหรับแก้ไขโปรแกรมต้นฉบับ โดยจะกลับไปสู่สถานะแก้ไขโปรแกรมต้นฉบับก่อนที่จะมีการสั่งคอมไพล์ตรวจสอบความถูกต้อง มีขั้นตอนดังนี้

- เลือกไปที่เมนูหลัก “Simulate” และคลิกเมนู “Edit Source Code” ดังรูปที่ ง-15

ภาคผนวก จ
วิธีการติดตั้งเครื่องมือ

วิธีการติดตั้งเครื่องมือ OAS Simulation

ความต้องการเบื้องต้น

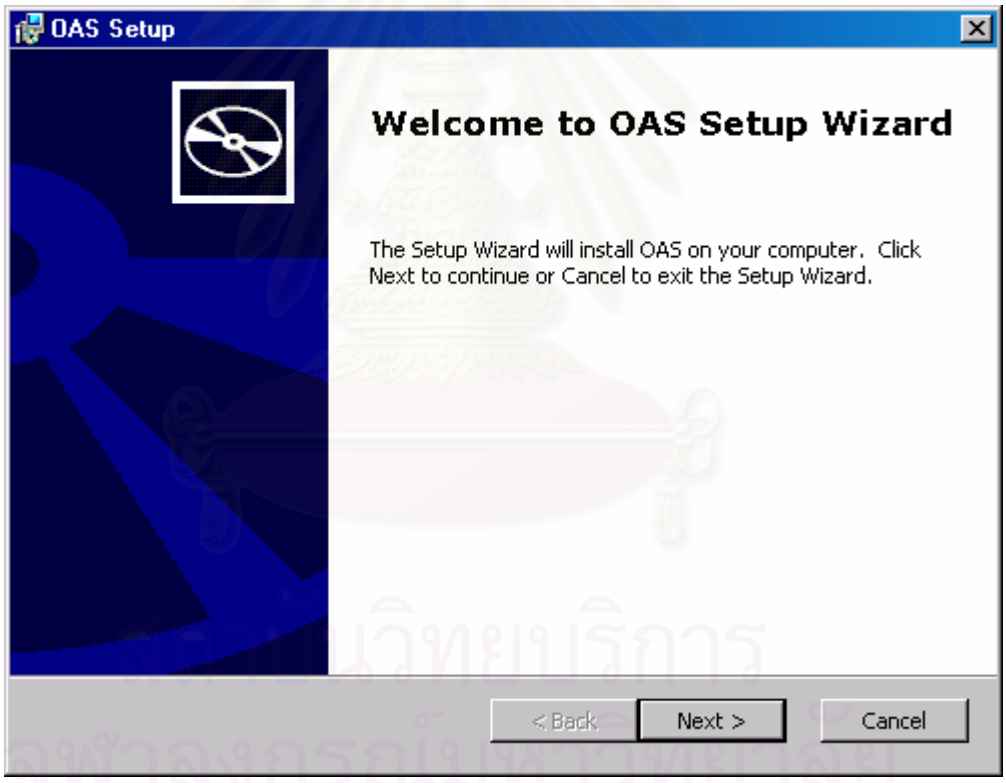
สิ่งที่ต้องการในการติดตั้งระบบมีดังต่อไปนี้

1. Windows 98, Windows XP Professional

2. พื้นที่ประมาณ 5 เมกะไบต์

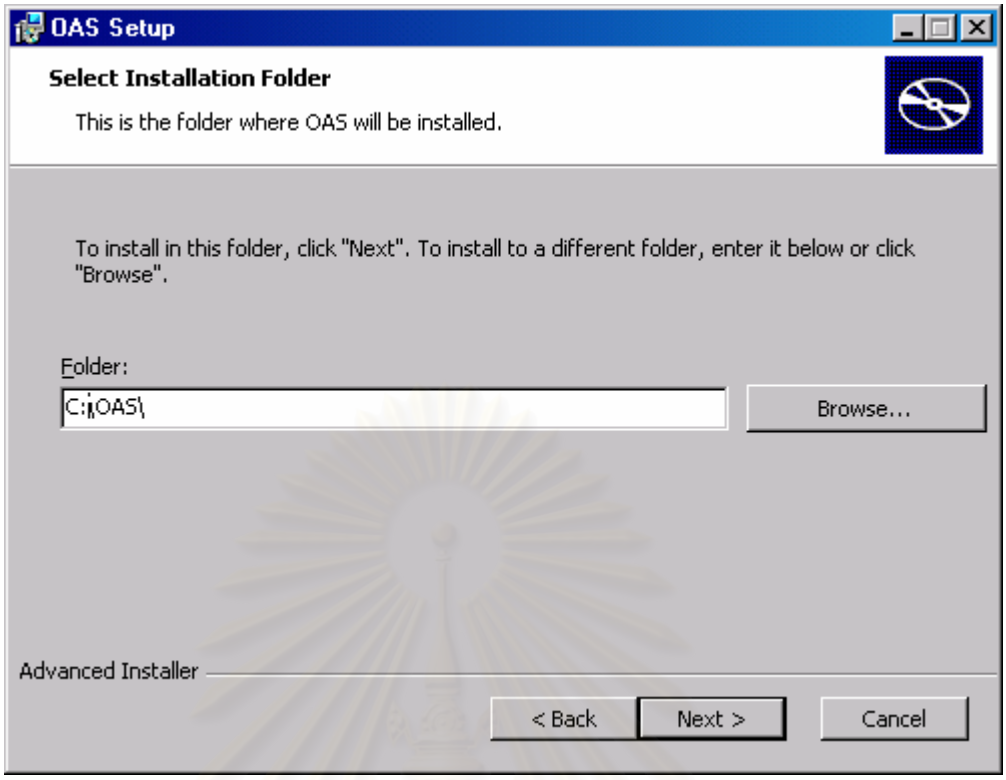
วิธีการติดตั้ง

1. ให้เรียกโปรแกรมชื่อว่า OAS-Installer.msi จากแผ่นซีดีรอมของงานวิจัยฉบับนี้ ที่ ..\Program\Setup จากนั้นโปรแกรม Setup จะแสดงหน้าจอการติดตั้งโปรแกรม ดังรูปที่ จ-1

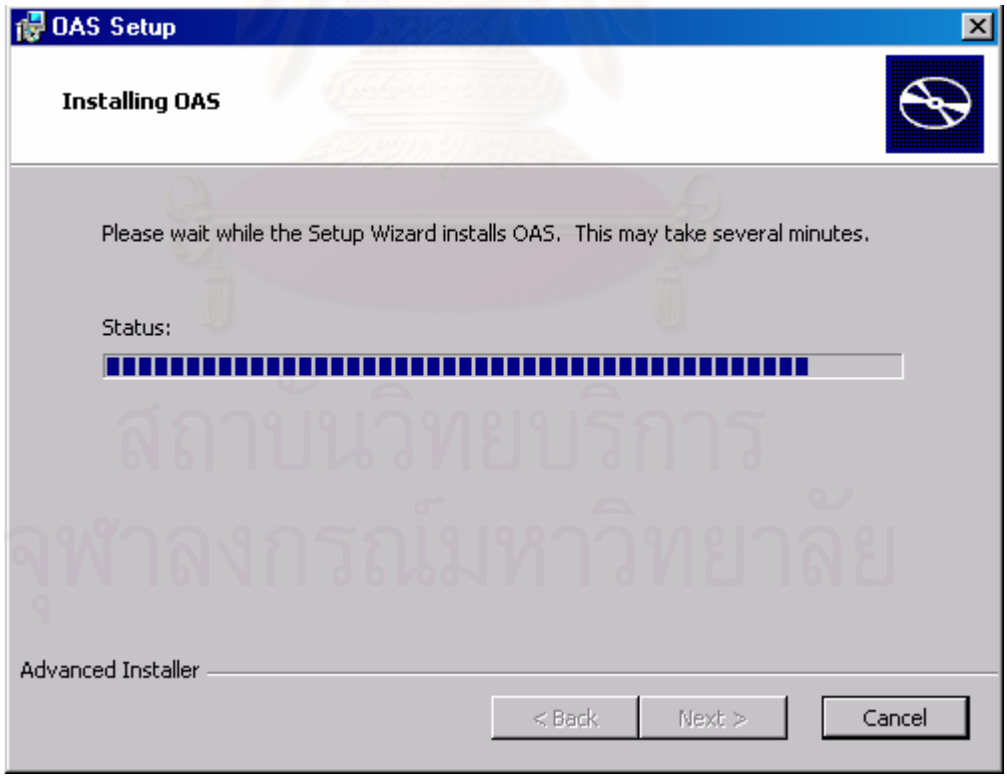


รูปที่ จ-1 แสดงหน้าจอการติดตั้ง

2. จากนั้นให้คลิกที่ปุ่ม Next เพื่อไปยังหน้าจอถัดไป จากนั้นให้ผู้เลือก Folder ที่ต้องการติดตั้งโปรแกรม ดังรูปที่ จ-2 เมื่อเลือกได้แล้วให้คลิกที่ปุ่ม Next อีกครั้ง แล้วทำการคลิกปุ่ม Next อีกครั้งระบบก็จะเริ่มทำการติดตั้งโปรแกรม ดังรูปที่ จ-3



รูปที่ จ-2 แสดงหน้าจอเลือก Folder ที่ต้องการติดตั้งโปรแกรม

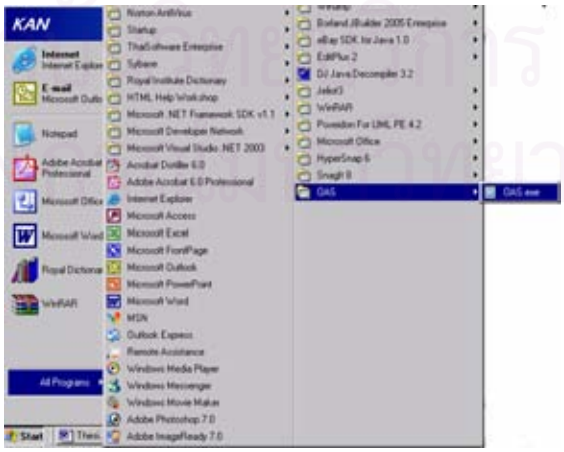


รูปที่ จ-3 หน้าจอแสดงสถานะการติดตั้งโปรแกรม

3. เมื่อโปรแกรมติดตั้งเรียบร้อยแล้ว ให้คลิกที่ปุ่ม Finish เพื่อปิดโปรแกรม ดังรูปที่ จ-4

รูปที่ จ-4 แสดงหน้าจอการติดตั้งโปรแกรมเสร็จเรียบร้อย

4. จากนั้นผู้ใช้สามารถเรียกใช้โปรแกรมได้ โดยเลือกที่ไอคอน  หรือเลือกที่ เมนู Start > All Programs > OAS และเลือก OAS.exe ดังรูปที่ จ-5



รูปที่ จ-5 แสดงวิธีการเรียกใช้โปรแกรม

ประวัติผู้เขียนวิทยานิพนธ์

นายกานต์ ปุริสชาติ เกิดวันที่ 28 สิงหาคม พ.ศ. 2518 จังหวัดตรัง สำเร็จการศึกษาระดับปริญญาวิทยาศาสตรบัณฑิต สาขาวิชาวิทยาการคอมพิวเตอร์ คณะเทคโนโลยีสารสนเทศ สถาบันราชภัฏสวนดุสิต ในปีการศึกษา 2541 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิชาวิทยาศาสตร์คอมพิวเตอร์ (ภาคนอกเวลาราชการ) คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2546 ปัจจุบันทำงานที่ธนาคาร BNP PARIBAS BANGKOK BRANCH ตำแหน่งผู้ช่วยผู้จัดการแผนกเทคโนโลยีสารสนเทศ