การดึงความรู้บนโค้ดภาษาเก่าแก่อาร์พีจีแสดงเป็นผังงาน

นางสาวกชพร สันติปารคู

วิทยานิพนธ์นี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญาวิทยาศาสตรมหาบัณฑิต
สาขาวิชาวิศวกรรมซอฟต์แวร์ ภาควิชาวิศวกรรมคอมพิวเตอร์
คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย
ปีการศึกษา 2556

FLOWCHART KNOWLEDGE EXTRACTION ON RPG LEGACY CODE

Miss Kochaporn Suntiparakoo

A Thesis Submitted in Partial Fulfillment of the Requirements

for the Degree of Master of Science Program in Software Engineering

Department of Computer Engineering

Faculty of Engineering

Chulalongkorn University

Academic Year 2013

หัวข้อวิทยานิพนธ์	การดึงความรู้บนโค้ดภาษาเก่าแก่อาร์พีจีแสดงเป็นผังงาน
โดย	นางสาวกชพร สันติปารคู
สาขาวิชา	วิศวกรรมซอฟต์แวร์
อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก	รองศาสตราจารย์ ดร. ญาใจ ลิ่มปิยะกรณ์

คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย อนุมัติให้นับวิทยานิพนธ์ฉบับนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรปริญญามหาบัณฑิต

.....คณบดีคณะวิศวกรรมศาสตร์
(ศาสตราจารย์ ดร. บัณฑิต เอื้ออาภรณ์)

คณะกรรมการสอบวิทยานิพนธ์

.....ประธานกรรมการ
(ผู้ช่วยศาสตราจารย์ ดร. สุกรี สินธุภิญโญ)

.....อาจารย์ที่ปรึกษาวิทยานิพนธ์หลัก
(รองศาสตราจารย์ ดร. ญาใจ ลิ่มปิยะกรณ์)

.....กรรมการภายนอกมหาวิทยาลัย
(อาจารย์ ดร. ภาสกร อภิรักษ์วรพินิต)

กชพร สันติปารคู : การดึงความรู้บนโค้ดภาษาเก่าแก่อาร์พีจีแสดงเป็นผังงาน. (FLOWCHART KNOWLEDGE EXTRACTION ON RPG LEGACY CODE) อ.ที่ปรึกษาวิทยานิพนธ์หลัก: รศ. ดร. ญาใจ ลิ่มปิยะกรณ์, 71 หน้า.

ซอฟต์แวร์ที่พัฒนาสืบทอดมานานมีลักษณะเป็นซอฟต์แวร์เก่าที่ยังคงให้บริการสำคัญหลักๆแก่องค์กร แอปพลิเคชันที่เขียนด้วยภาษาอาร์พีจีจัดได้ว่าเป็นซอฟต์แวร์เก่าที่พัฒนามานานเริ่มแรกอาร์พีจีถูกพัฒนาขึ้นเพื่อให้เป็นโปรแกรมสร้างรายงานโดยบริษัทไอบีเอ็ม แอปพลิเคชันธุรกิจมากมายถูกพัฒนาขึ้นด้วยภาษาอาร์พีจีและยังคงมีความสำคัญยิ่งยวดต่อการปฏิบัติงานของวิสาหกิจ หลังจากผ่านการใช้งานมาหลายทศวรรษ ระบบเก่าอาร์พีจีเหล่านี้ประสบปัญหาความยากในการบำรุงรักษา การปรับปรุงให้ดีขึ้น และการขยายความสามารถ เนื่องจากการขาดความเข้าใจระบบ และการจัดทำเอกสารที่อาจไม่มีความเป็นปัจจุบัน อันเป็นผลมาจากการเปลี่ยนแปลงทั้งหลายที่เกิดขึ้นกับซอฟต์แวร์ งานวิจัยนี้จึงได้นำเสนอวิธีการวิศวกรรมย้อนกลับเพื่อกู้คืนจุดประสงค์รหัสต้นทางอาร์พีจีที่พัฒนาสืบทอดมานาน เมตาเดตาถูกดึงรวบรวมจากอินพุตรหัสต้นทางอาร์พีจีโดยการตรวจจับและจัดการส่วนควบคุมและตัวดำเนินการโปรแกรม เมตาเดตาเหล่านี้จะถูกจัดเก็บในแผนภูมิแบบมีทิศทาง ซึ่งจะถูกเทียบไปเป็นรูปแบบภาษามาร์กอัปเพื่อแสดงผลเป็นภาพผังงานด้วยเครื่องมือการสร้างภาพกราฟวิซ ระบบต้นแบบที่พัฒนาขึ้นในงานนี้จะช่วยอำนวยความสะดวกในขั้นตอนการทำความเข้าใจโค้ดภาษาเก่าอาร์พีจีระหว่างกระบวนการบำรุงรักษาซอฟต์แวร์

ภาควิชา วิศวกรรมคอมพิวเตอร์ ลายมือชื่อนิสิต .....

สาขาวิชา วิศวกรรมซอฟต์แวร์ ลายมือชื่อ อ.ที่ปรึกษาวิทยานิพนธ์หลัก .....

ปีการศึกษา 2556

5570963721 : MAJOR SOFTWARE ENGINEERING
KEYWORDS: REVERSE ENGINEERING / LEGACY SYSTEM / RPG LANGUAGE / SOFTWARE MAINTENANCE / METADATA

KOCHAPORN SUNTIPARAKOO: FLOWCHART KNOWLEDGE EXTRACTION ON RPG LEGACY CODE. ADVISOR: ASSOC. PROF. YACHAI LIMPIYAKORN, 71 pp.

Legacy software can be characterized as old software that continues to provide core services to an organization. Applications written in RPG can be considered as legacy software. RPG was originated as a report-building program developed by IBM. Many business applications are written in RPG, and they are often critical in the operations of enterprises. Through decades of use, these RPG legacy systems can be hard to maintain, improve, and expand, since there is a general lack of understanding of the systems. The supporting documentation may not be current as well due to many changes implemented into the software. This paper thus presents a method of reverse engineering for recovering the intent of code from RPG legacy source. The metadata is gathered from the input RPG source by detecting and handling the program controls and operations. These metadata stored in the directed graph will then be mapped to DOT markup language format for flowchart rendering using visualization tool, Graphviz. The prototype implemented in this work would facilitate the understanding of RPG legacy code during software maintenance process.

Department: Computer Engineering Student's Signature .....
Field of Study: Software Engineering Advisor's Signature .....
Academic Year: 2013

กิตติกรรมประกาศ

วิทยานิพนธ์ฉบับนี้สำเร็จลุล่วงได้ด้วยดีจากความช่วยเหลือและสนับสนุนของบุคคลหลายท่าน ประกอบด้วย รองศาสตราจารย์ ดร.ญาใจ ลิ่มปิยะกรณ์ อาจารย์ที่ปรึกษาวิทยานิพนธ์ ซึ่งเป็นผู้เสียสละเวลาในการแนะแนวทางพัฒนา ชี้ให้เห็นถึงปัญหา และคอยแก้ไขตรวจสอบความเรียบร้อยของงานมาโดยตลอด และคณะกรรมการสอบวิทยานิพนธ์ ประกอบด้วยผู้ช่วยศาสตราจารย์ ดร.สุกรี สินธุภิญโญ และ อาจารย์ ดร.ภาสกร อภิรักษ์วรพินิต ซึ่งเป็นผู้ให้คำแนะนำและชี้จุดบกพร่องที่ควรแก้ไข ข้าพเจ้าจึงขอกราบขอบพระคุณเป็นอย่างยิ่งในความกรุณาของทุกท่านไว้ ณ ที่นี้

ท้ายที่สุด ข้าพเจ้าขอกราบขอบพระคุณ คุณพ่อ คุณแม่ และครอบครัว สำหรับกำลังใจที่มีค่ายิ่ง รวมถึงขอขอบพระคุณนิสิตรุ่นพี่สาขาวิศวกรรมซอฟต์แวร์ และมิตรสหายที่ให้กำลังใจ ให้การสนับสนุนและความช่วยเหลือในด้านต่างๆ และท่านอื่นๆ ที่มิได้กล่าวชื่อไว้ ณ ที่นี้ที่มีส่วนช่วยให้วิทยานิพนธ์ของข้าพเจ้าสำเร็จไปได้ด้วยดี

สารบัญ

หน้า

หน้า

หน้า

หน้า

สารบัญตาราง

หน้า

สารบัญภาพ

หน้า

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของปัญหา

ในปัจจุบันได้มีการพัฒนาซอฟต์แวร์เพื่อใช้ในการจัดการองค์กรเป็นจำนวนมาก ซึ่งหลังจากเสร็จสิ้นการพัฒนาซอฟต์แวร์จำเป็นต้องมีการบำรุงรักษาซอฟต์แวร์ (software maintenance) เพื่อเพิ่มคุณภาพของซอฟต์แวร์ ได้แก่ การตรวจสอบและแก้ไขข้อผิดพลาดในการทำงานของซอฟต์แวร์หลังการส่งมอบ หรือปรับปรุงซอฟต์แวร์เมื่อเกิดการเปลี่ยนแปลงความต้องการทางธุรกิจ เพื่อทำให้ซอฟต์แวร์ตอบสนองต่อความต้องการทางธุรกิจที่เป็นปัจจุบัน ทั้งนี้ ในขั้นตอนการบำรุงรักษาซอฟต์แวร์ ผู้บำรุงรักษาซอฟต์แวร์จำเป็นต้องเข้าใจการทำงานเดิมของซอฟต์แวร์ ด้วยการศึกษาและทำความเข้าใจตรรกะการทำงานของซอสโค้ดเดิม เพื่อทำให้การแก้ไขหรือการปรับปรุงซอฟต์แวร์เป็นไปอย่างถูกต้อง อย่างไรก็ตาม เอกสารการออกแบบหรือแผนภาพที่อธิบายขั้นตอนการทำงานของซอสโค้ดมักไม่เป็นปัจจุบัน โดยเฉพาะอย่างยิ่ง สำหรับระบบเก่าแก่ (Legacy System) ที่ยังสามารถใช้งานได้อยู่ อาจไม่สามารถหาเอกสารการออกแบบของระบบได้ ทำให้การบำรุงรักษาระบบมีความยากลำบาก ต้องใช้ระยะเวลาและความพยายามเป็นอย่างมากในการทำความเข้าใจซอสโค้ดของระบบเก่าแก่

ภาษาอาร์พีจี (RPG: Report Program Generator) เป็นภาษาโปรแกรมที่ใช้ในการพัฒนาซอฟต์แวร์ทางธุรกิจพัฒนาโดยบริษัทไอบีเอ็ม (IBM) เป็นระยะเวลายาวนานตั้งแต่ปี 1959 ซอฟต์แวร์ที่พัฒนาด้วยภาษาอาร์พีจีที่ยังใช้งานกันอยู่ในปัจจุบันเป็นซอฟต์แวร์ที่มีความสำคัญต่อองค์กร ซึ่งมีความยุ่งยากและมีความเสี่ยงสูงในการเปลี่ยนแปลงเป็นภาษาอื่น ดังนั้น ระบบซอฟต์แวร์ที่พัฒนาด้วยภาษาอาร์พีจีส่วนใหญ่จะเน้นไปที่การบำรุงรักษาเพื่อแก้ไขข้อผิดพลาดของซอฟต์แวร์ และการปรับปรุงแก้ไขซอฟต์แวร์เดิมเพื่อตอบสนองการเปลี่ยนแปลงความต้องการทางธุรกิจ การบำรุงรักษาซอฟต์แวร์ระบบเก่าแก่ดังกล่าวจำเป็นต้องศึกษาขั้นตอนการทำงานของซอสโค้ดเดิมที่มีอยู่แล้ว เพื่อให้เข้าใจและป้องกันข้อผิดพลาดร้ายแรงที่จะเกิดขึ้นกับซอฟต์แวร์ อันเนื่องมาจากการปรับปรุงหรือการแก้ไขที่ขาดความเข้าใจการทำงานของระบบ

งานวิจัยนี้ได้ทำการศึกษาข้อกำหนด (Specification) และวากยสัมพันธ์ (Syntax) ของการพัฒนาซอสโค้ดด้วยภาษาอาร์พีจี เพื่อนำเสนอวิธีการและพัฒนาเครื่องมือสร้างผังงานแสดงขั้นตอนการทำงานจากซอสโค้ดภาษาอาร์พีจี ซึ่งเป็นหนึ่งในภาษาที่ใช้พัฒนาระบบเก่าแก่ที่ยังสามารถใช้งานได้อยู่ ผังงานที่สร้างขึ้นดังกล่าวสามารถใช้เป็นเอกสารการออกแบบระบบที่ช่วยให้ผู้บำรุงรักษาซอฟต์แวร์ทำความเข้าใจตรรกะการทำงานของซอสโค้ดได้ง่ายขึ้น มีผลให้การตรวจสอบและแก้ไข

ข้อบกพร่อง (defect) รวมไปถึงการปรับปรุงเปลี่ยนแปลงซอสโค้ดกระทำได้อย่างถูกต้องและมีประสิทธิภาพมากยิ่งขึ้น กล่าวคือ ลดระยะเวลา ความพยายาม และลดข้อผิดพลาดในขั้นตอนการวิเคราะห์ และการออกแบบของการบำรุงรักษาระบบซอฟต์แวร์เก่าแก่

1.2 วัตถุประสงค์ของงานวิจัย

เพื่อนำเสนอวิธีการและพัฒนาเครื่องมือสนับสนุนการบำรุงรักษาระบบเก่าแก่ที่พัฒนาด้วยภาษาอาร์พีจี โดยการดึงความรู้จากซอสโค้ดมาสร้างเป็นผังงานแสดงรายละเอียดขั้นตอนการทำงาน เพื่อลดระยะเวลา ความพยายามที่ใช้ในการทำความเข้าใจซอสโค้ด และการจัดทำเอกสารการออกแบบ รวมทั้งลดข้อบกพร่องในการแก้ไขหรือปรับปรุงระบบเก่าแก่

1.3 ขอบเขตการวิจัย

1. ซอฟแวร์ที่พัฒนาขึ้นสามารถสร้างภาษากำกับเพิ่มดอทของผังงานแสดงรายละเอียดขั้นตอนการทำงานจากซอสโค้ดภาษาอาร์พีจี
2. ซอฟแวร์ที่พัฒนาขึ้นสามารถนำเข้าภาษากำกับเพิ่มดอทของผังงานแสดงรายละเอียดขั้นตอนการทำงานสู่โปรแกรมจินตทัศน์เพื่อสร้างแผนภาพผังงาน

1.4 ข้อตกลงเบื้องต้น

1. งานวิจัยรองรับการสร้างผังงานแสดงรายละเอียดขั้นตอนการทำงานจากซอสโค้ดภาษาอาร์พีจี/400
2. ซอสโค้ดข้อมูลนำเข้าอยู่ในรูปแบบแฟ้มข้อความ (txt)
3. ผังงานที่ได้จากจากแปลงซอสโค้ดจะอยู่ในรูปของรูปภาพ .jpeg

1.5 ประโยชน์ที่คาดว่าจะได้รับ

ได้วิธีการและเครื่องมือที่ช่วยบำรุงรักษาระบบเก่าแก่ที่พัฒนาด้วยภาษาอาร์พีจี โดยการดึงความรู้ในซอสโค้ดมาสร้างผังงานแสดงรายละเอียดขั้นตอนการทำงาน โดยผังงานที่สร้างขึ้นจะช่วยลดระยะเวลา ความพยายามในการทำความเข้าใจ และการจัดทำเอกสารการออกแบบของซอสโค้ดภาษาอาร์พีจี

1.6 วิธีดำเนินงานวิจัย

1. ศึกษาและทำความเข้าใจโครงสร้าง คำสั่ง และการทำงานของภาษาอาร์พีจี
2. ศึกษาและทำความเข้าใจทฤษฎีที่เกี่ยวข้อง
3. ศึกษาและทำความเข้าใจงานวิจัยที่เกี่ยวข้อง

4. ศึกษาขั้นตอนการพัฒนาเครื่องมือในการสร้างผังงาน
5. วิเคราะห์และกำหนดระเบียบวิธีวิจัย
6. ออกแบบ ตั้งสมมติฐาน ที่เกี่ยวข้องกับงานวิจัย
7. พัฒนาระบบการแปลงซอสโค้ดภาษาอาร์พีจีเป็นผังงาน
8. ทดสอบ และประเมินผลงานวิจัย
9. สรุปผลงานวิจัย และนำผลที่ได้ไปปรับปรุงระบบเพื่อให้ได้วัตถุประสงค์ที่กำหนด
10. ตีพิมพ์ผลงานวิจัย
11. จัดทำรูปเล่มวิทยานิพนธ์

1.7 ลำดับการจัดเรียงเนื้อหาในวิทยานิพนธ์

วิทยานิพนธ์นี้แบ่งเนื้อหาออกเป็น 6 บท ดังต่อไปนี้ บทที่ 1 บทนำ กล่าวถึงความเป็นมาและความสำคัญของปัญหา วัตถุประสงค์ของการวิจัย ขอบเขตของการวิจัย ประโยชน์ที่คาดว่าจะได้รับ วิธีดำเนินงานวิจัย และผลงานตีพิมพ์ บทที่ 2 กล่าวถึงทฤษฎี และงานวิจัยที่เกี่ยวข้อง บทที่ 3 กล่าวถึงวิธีดำเนินงานวิจัย บทที่ 4 กล่าวถึงการออกแบบ และพัฒนาระบบตามแนวทางการวิจัย บทที่ 5 กล่าวถึงวิธีการประเมินและวัดผลการทดลอง และบทที่ 6 สรุปผลการวิจัย ข้อเสนอแนะ และแนวทางสำหรับการวิจัยต่อในอนาคต

1.8 ผลงานที่ตีพิมพ์จากวิทยานิพนธ์

ส่วนหนึ่งของวิทยานิพนธ์นี้ได้รับการตีพิมพ์ในรายงานสืบเนื่องจากการประชุมวิชาการระดับนานาชาติ เรื่อง “Flowchart Knowledge Extraction on RPG Legacy Code”, Kochaporn Suntiparakoo and Yachai Limpiyakorn, in Proceedings of 2013 International Conference on Advanced Software Engineering & Its Applications, Jeju Island, Korea, Nov 21-23, 2013, pp. 258-263., และตีพิมพ์ในวารสารวิชาการนานาชาติ เรื่อง “Recovering Intent of Code from RPG Legacy Source”, Kochaporn Suntiparakoo and Yachai Limpiyakorn, International Journal of Software Engineering and Its Applications, vol. 3,no. 4, (2014), pp. 291–304

บทที่ 2

ทฤษฎีและงานวิจัยที่เกี่ยวข้อง

2.1 ทฤษฎีที่เกี่ยวข้อง

2.1.1 ภาษาอาร์พีจี (RPG Language: Report Program Generator Language) [1]

ภาษาอาร์พีจีถูกพัฒนาขึ้นตั้งแต่ปี ค .ศ.1959 เป็นภาษาโปรแกรมที่ใช้ในการพัฒนาซอฟต์แวร์ทางธุรกิจ โดยมีวัตถุประสงค์เริ่มแรกเป็นโปรแกรมเพื่อออกรายงาน ซึ่งถูกใช้กับระบบคอมพิวเตอร์เซิร์ฟเวอร์ของไอบีเอ็ม

ภาษาอาร์พีจีเป็นภาษาโปรแกรมเชิงโครงสร้าง รูปแบบการเขียนโปรแกรมจะอยู่ในรูปแบบ 1 คำสั่งการทำงานต่อหนึ่งบรรทัด ในการพัฒนาภาษาอาร์พีจีจะต้องคำนึงถึงตำแหน่งของคำสั่งการทำงาน ซึ่งตำแหน่ง และคำสั่งของการทำงานจะขึ้นอยู่กับ Specification ที่กำหนดโดยการกำหนด Specification ของโปรแกรมจะอยู่ในตำแหน่งแรกของทุกบรรทัดคำสั่ง

ภาษาอาร์พีจีประกอบด้วย 7 Specification ในการพัฒนาต้องมีการเรียงลำดับ Specification ดังนี้

- Control Specification (H) เป็นส่วนการใส่ข้อมูลที่เกี่ยวกับโปรแกรม (เป็น optional สามารถละได้)
- File description specification (F) เป็นส่วนของการกำหนดไฟล์ทั้งหมดที่ใช้ในโปรแกรม
- Extension specification (E) เป็นส่วนกำหนดตาราง และอาร์เรย์ ที่ใช้ในโปรแกรม
- Line counter specification (L) เป็นส่วนที่ใช้ในการอธิบายเลขหน้าของการออกรายงาย
- Input specification (I) เป็นส่วนของการกำหนดค่าคงที่ โครงสร้างของข้อมูลที่ใช้ภายในโปรแกรม
- Calculation specification (C) เป็นส่วนที่ใช้ในการคำนวณ และประมวลผล
- Output specification (O) เป็นส่วนที่ใช้ในการกำหนดข้อมูลของไฟล์ที่เป็นเอาต์พุต

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
FFilenameIPEAF....RlenLK1AIOvKlocEDevice+......KExit++Entry+A....U1.*
FPRG01FM CF  E                    WORKSTN
FEMPMST  UF  E           K        DISK                      A
FPRJMST  UF  E           K        DISK                      A
FRSNMST  UF  E           K        DISK                      A
```

ภาพที่ 1 ตัวอย่างการเขียนโปรแกรมอาร์พีจีในส่วนของ File description specification

ภาพที่ 1 แสดงตัวอย่างการเขียนโปรแกรมอาร์พีจีในส่วนของการประกาศไฟล์ซึ่งอยู่ในส่วนของ File description specification โดยตำแหน่งที่ 6 ของทุกบรรทัดคำสั่งถูกกำหนดเป็น “F”เพื่อระบุว่าเป็นส่วนของ File description specification ตัวอย่างการประกาศไฟล์ที่ใช้ในโปรแกรม จากภาพที่ 1 ได้แก่ ไฟล์ PRG01FM เป็นหน้าจอแสดงผลการทำงาน (display file), ไฟล์ EMPMST, PRJMST, RSNMST ซึ่งเป็นไฟล์ที่สามารถอัพเดทข้อมูลได้

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
E*  Compile time array containing error descriptions.
E....FromfileTofile++Name++N/rN/tbLenPDSArrnamLenPDSComments++++++++*
E                    ERR     1  10 50
```

ภาพที่ 2 ตัวอย่างการเขียนโปรแกรมอาร์พีจีในส่วนของ Extension specification

ภาพที่ 2 แสดงตัวอย่างการประกาศอาร์เรย์ ชื่อว่าERR ซึ่งอยู่ในส่วนของ Extension specification โดยตำแหน่งที่ 6 ของทุกบรรทัดคำสั่งถูกกำหนดเป็น “E”

```
*.. 1 ...+... 2 ...+... 3 ...+... 4 ...+... 5 ...+... 6 ...+... 7 ..*
CL0N01N02N03Factor1+++OpcdeFactor2+++ResultLenDHHiLoEqComments++++++*
C           BEGIN     TAG
C                     MOVE '0'       *IN60
C                     MOVE *BLANKS   EMESS
C                     MOVE *BLANKS   EMPAPL
C                     MOVE *BLANKS   PRJAPL
C                     MOVE *BLANKS   RSNAPL
```

ภาพที่ 3 ตัวอย่างการเขียนโปรแกรมอาร์พีจีในส่วนของ Calculation specification

ภาพที่ 3 แสดงตัวอย่างการกำหนดค่าว่างให้กับตัวแปร EMESS, EMPAPL, PRJAPL, RSNAPL ซึ่งอยู่ใน Calculation specification โดยตำแหน่งที่ 6 ของทุกบรรทัดคำสั่งถูกกำหนดเป็น “C” ในการพัฒนาผังงานแสดงขั้นตอนการทำงานของภาษาอาร์พีจีจะใช้ Calculation Specification เป็นหลักในการพัฒนา เนื่องมาจากเป็น ส่วนสำคัญที่ใช้ในการคำนวณ และการประมวลผลของโปรแกรม ซึ่งรายละเอียดตำแหน่งของ Calculation Specification แสดงในตารางที่ 1

ตารางที่ 1 รายละเอียดตำแหน่งของ Calculation Specification

Position	Argument Type	Description
6	Form Type	A ‘C’ must appear in position 6 to identify this line as a calculation specification statement.
7	Comment	An ‘*’ must appear in position 7 to

		identify this line as a header comment.
9-17	Indicator	Positions 10 and 11, 13 and 14, and 16 and 17 contain indicators that are tested to determine if a particular calculation is to be processed. A blank in positions 9, 12, and 15 designates that the indicator must be on for a calculation to be done. A N in positions 9, 12, and 15 designates that the associated indicator must be off for a calculation to be done.
18-27	Factor 1	The entries that are valid for factor 1 depend on the operation code specified in positions 28 through 32. For the specific entries for factor 1 for a particular operation code.
28-32	Operation Code	Operation to be executed using factor 1, factor 2, and the result field entries.
33-42	Factor 2	The entries that are valid for factor 2 depend on the operation code specified in positions 28 through 32. For the file operation codes, factor 2 names a file or record format to be used.
43-48	Result Field	The result field names the field that contains the result of the calculation operation specified in positions 28 through 32.
49-51	Field Length	Specify the length of the result field. This entry is optional, but can be used to define a field not defined elsewhere

		in the program.
52	Decimal Position	Position 52 indicates the number of positions to the right of the decimal in a numeric result field. If the numeric result field contains no decimal positions, enter a '0' (zero). This position must be blank if the result field is character data.
53	Operation Extender	The operation extenders are single-character entries that provide additional attributes to the operations that they accompany.
54-59	Resulting Indicators	These positions can be used, for example, to test the value of a result field after the completion of an operation, or to indicate an end-of-file, error, or record-not-found condition. The resulting indicator positions designate different uses, depending on the operation code specified.
60-74	Comment	Comment in line with display when print out.
75-80	Comment	Comment in line without display when print out.

2.1.2 การควบคุมการทำงานในโปรแกรมอาร์พีจี/400 (Control in the RPG/400 Programming Language) [1]

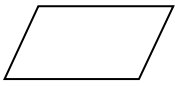
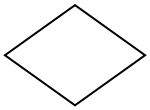
การควบคุมการทำงานในโปรแกรมอาร์พีจี/400 ประกอบไปด้วย

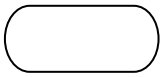
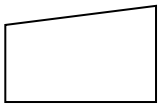
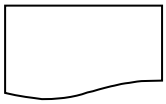
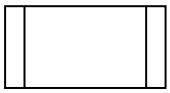
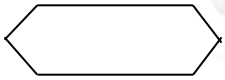
- การทำงานแบบลำดับ (Sequential operation)
- การทำงานแบบแยกตามเงื่อนไข (Conditional branching) ได้แก่
    - If else Structure
    - Select Structure
    - GOTO operation
    - Execute subroutine
    - The CASXX (Compare and Branch) operation
    - The CABXX (Conditionally Invoke Subroutine) operation
- การดำเนินการซ้ำตามเงื่อนไข (Repeating an operation based on a certain condition) ได้แก่
    - Do operation
    - Do while operation
    - Do until operation

2.1.3 ผังงาน (Flowchart)

ผังงาน เป็นแผนภาพแสดงลำดับขั้นตอนของการทำงานแต่ละขั้นตอน ซึ่งแสดงโดยการใช้สัญลักษณ์ที่มีความหมายบ่งบอกลักษณะการทำงานแต่ละประเภท ขั้นตอนแต่ละขั้นตอนจะถูกเชื่อมด้วยลูกศร เพื่อบ่งบอกถึงลำดับการทำงานตามทิศทางของลูกศร สัญลักษณ์ที่ใช้ในการเขียนผังงานตามมาตรฐาน ANSI (American National Standards Institute) แสดงดังตารางที่ 2

ตารางที่ 2 สัญลักษณ์ที่ใช้ในการเขียนผังงานตามมาตรฐานANSI

สัญลักษณ์	ความหมาย
	การประมวลผลข้อมูล การกำหนดค่า การโยกย้ายข้อมูล หรือการคำนวณทางคณิตศาสตร์
	หน่วยรับ หรือแสดงผลข้อมูลโดยไม่ระบุอุปกรณ์
	กำหนดเงื่อนไขทางเลือก การเปรียบเทียบทางตรรกศาสตร์ เพื่อการตัดสินใจ

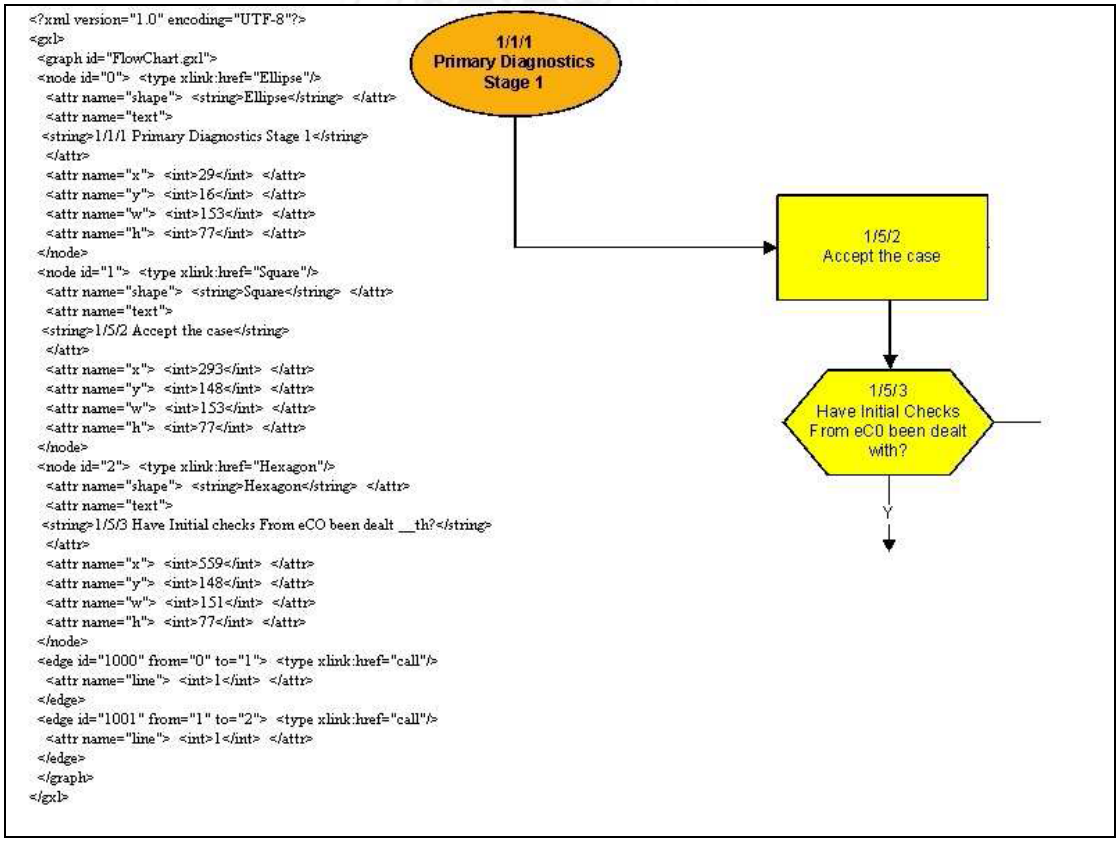
	แสดงจุดเริ่มต้น และจุดสิ้นสุด
	การป้อนข้อมูลเข้าทางแป้นพิมพ์
	การรับหรือแสดงผลข้อมูลทางแม่เหล็ก
	แสดงผลจอภาพ
	แสดงผลทางเครื่องพิมพ์
	แหล่งเก็บข้อมูล หน่วยความจำสำรอง
	โปรแกรมย่อย หรือโมดูล เริ่มทำงาน โดยหลังจากคำสั่งจากโปรแกรมย่อยแล้วจะกลับมาทำคำสั่งถัดไป
	การเตรียมทำงานลำดับถัดไป
	จุดเชื่อมต่อผังงานในหน้าเดียวกัน
	จุดเชื่อมต่อผังงานที่อยู่ คนละหน้า
	เส้นเชื่อม และหัวลูกศรแสดงทิศทางการทำงานของผังงาน

2.2 งานวิจัยที่เกี่ยวข้อง

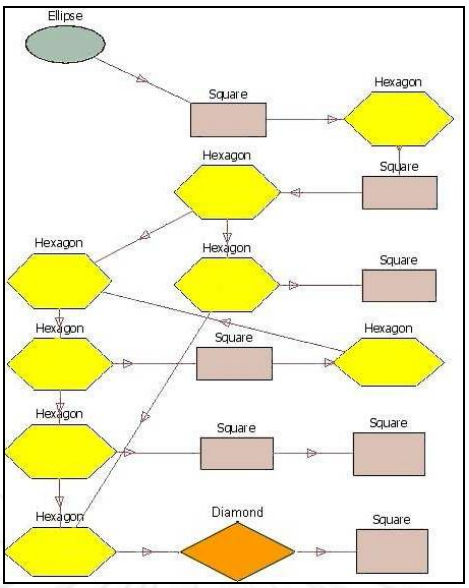
2.2.1 Flowchart Knowledge Extraction on Image Processing [2]

งานวิจัยนี้นำเสนอการดึงความรู้จากรูปภาพผังงานด้วยวิธีการประมวลผลภาพ (Image processing) ผลลัพธ์ที่ได้คือ เมตาเดตา (metadata) ที่อยู่ในรูปแบบเอ็กซ์เอ็มแอล (XML format) (ภาพที่ 4) ของภาษาจีเอ็กซ์แอล (GXL: Graph eXchange Language) [3] ซึ่งสามารถแปลง GXL ให้เป็นผังงานด้วยโปรแกรม SHriMP (ภาพที่ 5) ข้อดีของเมตาเดตา คือ สามารถใช้ในการแชร์ข้อมูล โดยให้รายละเอียดและโครงสร้างที่อยู่บนมาตรฐานเดียวกัน รวมทั้งจัดสร้างเป็นฐานความรู้ (Knowledge Base) เพื่อเป็นสถานที่เก็บเอกสารสำคัญ (archive)

จีเอ็กซ์แอลคือ รูปแบบมาตรฐานในการแลกเปลี่ยนข้อมูล ซึ่งมีไวยากรณ์ในรูปแบบภาษาเอ็กซ์เอ็มแอล โดยจีเอ็กซ์แอลจะให้รายละเอียดในเรื่องประเภท (type) คุณลักษณะ (attribute) ทิศทาง (directed) และลำดับ (ordered) ของกราฟ ซึ่งสามารถแสดงอยู่ในรูปของ hypergraphs และ hierarchical graph [3]



ภาพที่ 4 ตัวอย่าง GXL และผังงานที่แสดงผ่านโปรแกรม Visualization

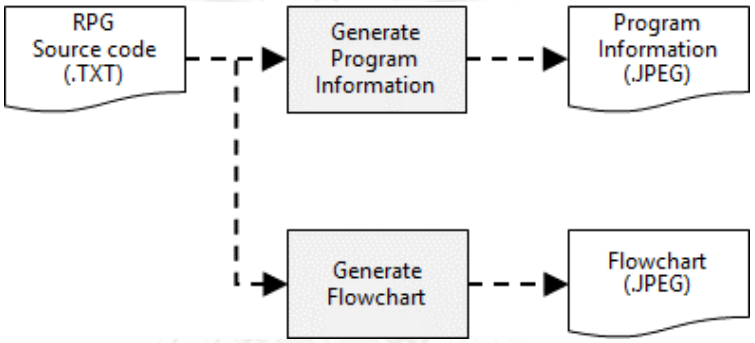


ภาพที่ 5 ผลลัพธ์ผังงานที่แสดงผ่านโปรแกรม Visualization ด้วยการนำเข้า GXL

บทที่ 3
วิธีดำเนินการวิจัย

3.1 แนวคิดวิธีดำเนินการวิจัย

งานวิจัยฉบับนี้เป็นการนำเสนอการพัฒนาเครื่องมือสร้างผังงานแสดงขั้นตอนการทำงานของภาษาอาร์พีจี เพื่อนำมาใช้เป็นเอกสารประกอบการทำความเข้าใจในการบำรุงรักษา หรือแก้ไขปรับปรุงซอฟต์แวร์ให้ตอบสนองต่อการเปลี่ยนแปลงความต้องการทางธุรกิจ โดยมีการทำงานดังภาพที่ 6



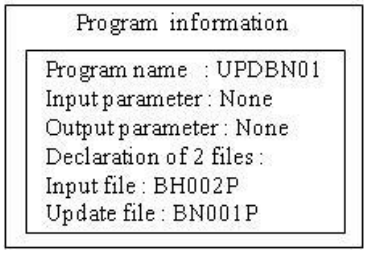
ภาพที่ 6 ภาพรวมการทำงานเครื่องมือสร้างผังงานแสดงขั้นตอนการทำงานของภาษาอาร์พีจี

จากภาพที่ 6 อธิบายการทำงานของเครื่องมือสร้างผังงานแสดงขั้นตอนการทำงานของภาษาอาร์พีจี จะมีข้อมูลนำเข้าเป็นซอสโค้ดภาษาอาร์พีจีที่อยู่ในรูปของเท็กซ์ไฟล์ (Text File) ซึ่งการทำงานจะประกอบด้วย 2 ส่วนหลักๆ ได้แก่ การสร้างข้อมูลของโปรแกรม (Generate Program Information) และการสร้างแผนภาพผังงาน (Generate Flowchart) โดยผลลัพธ์สุดท้ายจะได้เป็นแผนภาพ .JPEG

3.2 การสร้างข้อมูลของโปรแกรม (Program Information)

ข้อมูลของโปรแกรมเป็นส่วนสำคัญในการมองภาพรวมการทำงานของโปรแกรมเพื่อให้เห็นข้อมูลนำเข้า (Input) ข้อมูลส่งออก (Output) รวมทั้งไฟล์ที่เกี่ยวข้อง และไฟล์ที่ใช้ในโปรแกรมทั้งหมด ดังภาพที่ 7 ซึ่งข้อมูลของโปรแกรมที่สร้างขึ้นจะประกอบด้วย

- ชื่อโปรแกรม (Program Name)
- พารามิเตอร์นำเข้า (Input Parameter)
- พารามิเตอร์ส่งออก (Output Parameter)
- ไฟล์ที่เกี่ยวข้อง ได้แก่ ไฟล์นำเข้า (Input File) และไฟล์ส่งออก (Output File)

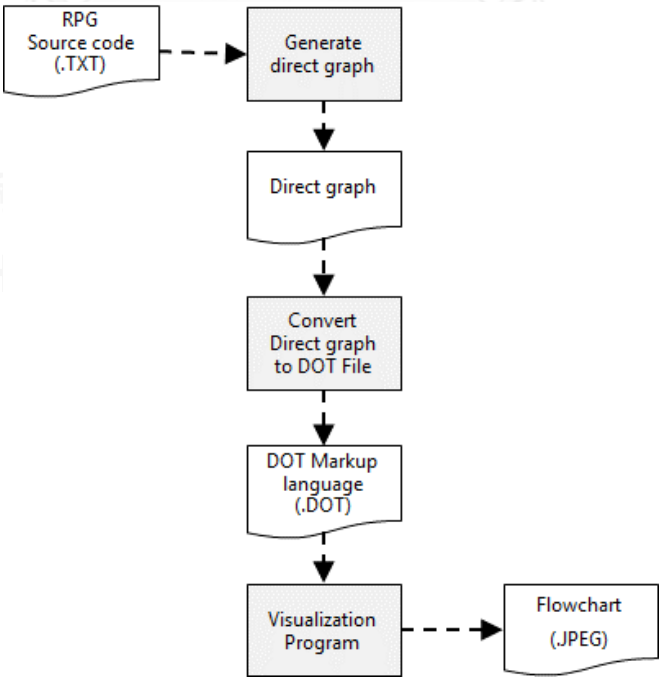


ภาพที่ 7 ตัวอย่างข้อมูลของโปรแกรม

ในการสร้างข้อมูลของโปรแกรมนั้นจะใช้ข้อมูลในส่วน File Description Specification และ Calculation Specification โดยที่พารามิเตอร์ส่งออก และพารามิเตอร์นำเข้าสามารถนำข้อมูลมาจาก Calculation Specification ซึ่งพารามิเตอร์ส่งออกจะมีการอัพเดทข้อมูลก่อนจบโปรแกรม และพารามิเตอร์นำเข้าจะไม่มีการอัพเดทข้อมูลก่อนจบโปรแกรม ในส่วนข้อมูลไฟล์นำเข้า และไฟล์ส่งออก สามารถนำข้อมูลจาก File Description Specification ซึ่งเป็นส่วนของการกำหนดไฟล์ทั้งหมดที่ใช้ในโปรแกรม

3.3 การสร้างแผนภาพผังงาน (Flowchart)

การสร้างแผนภาพผังงานเริ่มต้นจากการนำเข้าซอสโค้ดภาษาอาร์พีจีที่อยู่ในรูปของเท็กซ์ไฟล์ จากนั้นทำการดึงข้อมูลซอสโค้ดจัดให้อยู่ในรูปของกราฟแบบมีทิศทาง แล้วจึงแปลงข้อมูลจากกราฟแบบมีทิศทางให้อยู่ในรูปแบบภาษากำกับเพิ่มดอท เพื่อนำเข้าสู่โปรแกรมจินตทัศน์ โดยผลลัพธ์สุดท้ายจะได้เป็นแผนภาพผังงาน .JPEG ดังภาพที่ 8



ภาพที่ 8 ภาพรวมกระบวนการการสร้างแผนภาพผังงาน

3.3.1 คุณลักษณะกราฟแบบมีทิศทางที่ใช้เก็บข้อมูลซอสโค้ดภาษาอาร์พีจี

กราฟแบบมีทิศทางที่นำมาใช้เก็บข้อมูลของภาษาอาร์พีจีเพื่อทำการแปลงเป็นภาษากำกับดอท ประกอบด้วยโหนด (Node) และเส้นเชื่อมระหว่างโหนด (Edge) โดยที่โหนด และเส้นเชื่อมระหว่างโหนด จะมีคุณลักษณะดังต่อไปนี้

3.3.1.1 คุณลักษณะโหนดของกราฟแบบมีทิศทาง

คุณลักษณะโหนดของกราฟแบบมีทิศทางจะประกอบด้วยข้อมูลที่สำคัญในการสร้างโหนดของกราฟด้วยภาษากำกับเพิ่มดอทซึ่งประกอบไปด้วย

- ไอดี (ID) เป็นตัวแทนของแต่ละโหนด โดยที่ไอดีของแต่ละโหนดจะไม่ซ้ำกัน (Unique)
- ข้อความ (Text) เป็นข้อความที่ได้จากการแปลงซอสโค้ดภาษาอาร์พีจี
- ประเภทของโหนด(Type) เป็นประเภทรูปร่างของแต่ละโหนด ได้แก่ โหนดที่เป็นกระบวนการ (Process) และโหนดที่เป็นเงื่อนไข (Condition)

3.3.1.2 คุณลักษณะเส้นเชื่อมระหว่างโหนดของกราฟแบบมีทิศทาง

คุณลักษณะเส้นเชื่อมของกราฟแบบมีทิศทางจะประกอบด้วยข้อมูลที่สำคัญในการสร้างเส้นเชื่อมระหว่างโหนดด้วยภาษากำกับดอท ซึ่งประกอบไปด้วย

- โหนดต้นทาง (Source node)
- โหนดปลายทาง (Destination node)
- ข้อความ (Label) เป็นข้อความกำกับเส้นเชื่อมระหว่าง โหนดต้นทาง และโหนดปลายทาง

3.3.2 การสร้างกราฟแบบมีทิศทางของผังงานโดยละเอียดจากซอสโค้ดภาษาอาร์พีจี

ในการสร้างกราฟแบบมีทิศทางของผังงานแสดงขั้นตอนการทำงานของภาษาอาร์พีจีจะใช้ซอสโค้ดในส่วนของ Calculation specification เพื่อสร้างเป็นกราฟแบบมีทิศทางเป็นหลัก ซึ่งมีขั้นตอนดังต่อไปนี้

3.3.2.1 ตัดแบ่งซอสโค้ดในส่วนของ Calculation Specification

ในกระบวนการแรกเป็นการตัดแบ่งซอสโค้ดในส่วนของ Calculation Specification ตามตารางที่ 1 จากนั้นทำการตรวจสอบซอสโค้ดในส่วนของรหัสดำเนินการ (Operation Code) ซึ่งอยู่ในตำแหน่งที่ 28-32 เพื่อนำไปสร้างโหนด และเส้นเชื่อมระหว่างโหนดของกราฟแบบมีทิศทาง โดยแต่ละรหัสดำเนินการ จะมีการแปลงเป็นข้อความที่สามารถเข้าใจง่ายตามตารางที่ 3 ในขั้นตอนถัดไป

3.3.2.2 ตรวจสอบรหัสดำเนินการ (Detect operation code)

รหัสดำเนินการที่ได้จากการตัดแบ่งซอสโค้ดในตำแหน่งที่ 32-28 จะถูกตรวจสอบ เพื่อนำไปสร้างโหนด และเส้นเชื่อมของกราฟแบบมีทิศทาง โดยที่รหัสดำเนินการจะถูกแบ่งออกเป็น 2 ประเภท ได้แก่

1. รหัสดำเนินการที่เกี่ยวข้องกับ Resulting Indicator คือ รหัสดำเนินการที่มีเอาท์พุต (Output) ในส่วนของ Resulting Indicator

2. รหัสดำเนินการที่ไม่เกี่ยวข้องกับ Resulting Indicator คือ รหัสดำเนินการที่ไม่มี เอาท์พุต ในส่วนของ Result Indicator

สำหรับการขั้นตอนการตรวจสอบรหัสดำเนินการทุกๆ รหัสดำเนินการจะถูกสร้างเป็น โหนด ของกราฟแบบมีทิศทางโดยข้อความในแต่ละโหนดจะถูกแปลงให้อยู่ในรูปแบบของภาษาที่เข้าใจได้ง่าย ดังตัวอย่างตารางที่ 3 (รายละเอียดในภาคผนวก ก)

ตารางที่3 ข้อความของแต่ละรหัสดำเนินการ

รหัสดำเนินการ	**ข้อความ**
ADD	Case Factor 1 = Blank *[Result Field] = [Result Field] + [Factor 2]* Case Factor 1 <> Blank *[Result Field] = [Factor 1] + [Factor 2]*
CHAIN	Search File *[Factor 2]* by key *[Factor 1]*
DIV	*[Result Field] = [Factor 1] / [Factor 2]*
DOWEQ	IF *[Factor 1] = [Factor 2]*
DOUEQ	IF *[Factor 1] = [Factor 2]*
ELSE	Else
ENDDO	-
ENDIF	End If
ENDSL	-
IFEQ	IF (*[Factor 1] = [Factor 2]*) **Keep node in stack
IFGT	IF (*[Factor 1] > [Factor 2]*) **Keep node in stack

IFLT	IF (*[Factor 1]* < *[Factor 2]*) **Keep node in stack
KLIST	Define a composite key *[Factor 1]*
PLIST	Identify a parameter list *[Factor 1]*
READ	Read file *[Factor 2]*
SELEC	-
Z-ADD	*[Result Field]* = *[Factor 2]*

ในกรณีของรหัสดำเนินการที่มีเอาท์พุตในส่วนของ Resulting Indicator จะมีการเก็บข้อมูลของ Resulting Indicator เพื่อนำไปใช้ในกรณีที่มีการนำผลของ Resulting Indicator ในรหัสดำเนินการถัดไป

3.3.2.3 ตรวจสอบและแปลความหมาย Resulting Indicator

การตรวจสอบ Resulting Indicator สามารถตรวจสอบได้จากซอสโค้ดที่ถูกตัดแบ่งในตำแหน่งที่ 54-59 ซึ่งสามารถมี Resulting Indicator ได้สามตัว แบ่งเป็น ตำแหน่งที่ 54-55 ตำแหน่งที่ 56-57 และตำแหน่งที่ 58-59 โดยที่ความหมายผลลัพธ์ของแต่ละ Indicator จะแตกต่างกันขึ้นอยู่กับรหัสดำเนินการตามตารางที่ 4 แสดงความหมายผลลัพธ์ Resulting Indicator ของรหัสดำเนินการ CHAIN, READ, WRITE และ UPDATE ในกรณีที่มีสถานะเป็น ON และสถานะเป็น OFF

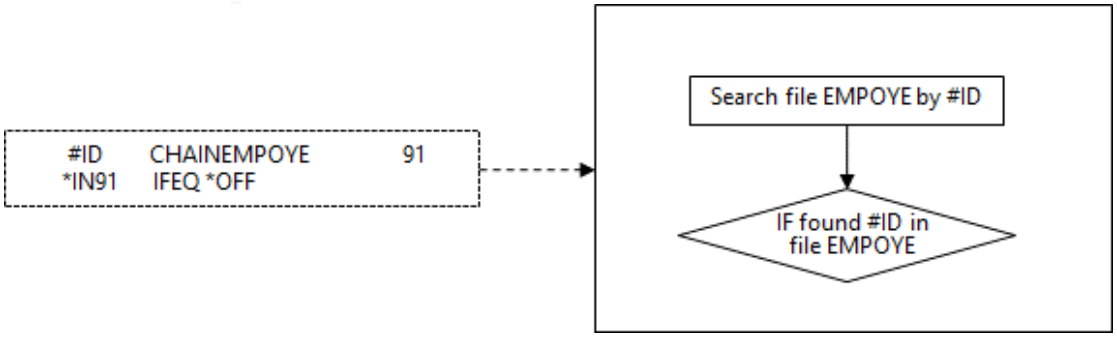
ตารางที่ 4 ความหมายของเอาต์พุตในส่วน Resulting Indicator ของรหัสดำเนินการ CHAIN, READ, WRITE และ UPDATE

รหัสดำเนินการ	**ตำแหน่งของ Indicator**	**ข้อความ**	
		เมื่อ Indicator มีสถานะเป็น ON	**เมื่อ Indicator มีสถานะเป็น OFF**
CHAIN	54-55	Search Key [Key Name] in file [Fine Name] Not found	Search Key [Key Name] in file [Fine Name] found
	56-57	Search operation Key [Key Name] in file [Fine Name] not complete	Search operation Key [Key Name] in file [Fine Name] complete
	58-59	-	-

READ	54-55	-	-
	56-57	Read file [File Name] complete	Read file [File Name] not complete
	58-59	Read file [File Name] end of file	Read file [File Name] not end of file
WRITE	54-55	-	-
	56-57	Write file [File Name] complete	Write file [File Name] not complete
	58-59	-	-
UPDATE	54-55	-	-
	56-57	Update file [File Name] not complete	Update file [File Name] complete
	58-59	-	-

3.3.2.4 แปลงซอสโค้ดเป็นข้อความของโหนด

การแปลงซอสโค้ดเป็นข้อความของโหนดของกราฟแบบมีทิศทางเริ่มต้นจากการตรวจสอบรหัสดำเนินการ ในกรณีที่เป็นรหัสดำเนินการที่ไม่มีการเปรียบเทียบสถานะของ Indicator จะทำการแปลความหมายตามตารางที่ 3 ในกรณีที่เป็นรหัสดำเนินการที่มีการเปรียบเทียบสถานะของ Indicator จะทำการแปลความหมายตามตารางที่ 4 ซึ่งข้อความจะขึ้นอยู่กับ สถานะของ Indicator ที่เกี่ยวข้อง



ภาพที่ 9 ตัวอย่างการแปลความหมายของรหัสดำเนินการที่มีการเปรียบเทียบสถานะของ Indicator

ภาพที่ 9 ตัวอย่างการแปลความหมายของรหัสดำเนินการที่มีการเปรียบเทียบสถานะของ Indicator จะเห็นว่ารหัสดำเนินการ CHAIN มี Resulting Indicator ตำแหน่งที่ 54-55 คือ 91 ซึ่งเป็น Indicator ที่ใช้ในการตรวจสอบการค้นหาเรคคอร์ด (Record) ที่มีคีย์ (Key) เป็น #ID ในไฟล์ EMPOYE หากผลลัพธ์ของ Indicator 91 เป็น ON แสดงว่าการค้นหาไม่พบเรคคอร์ดที่มีค่าคีย์เป็น #ID หากผลลัพธ์ของ indicator 91 เป็น OFF แสดงว่าสามารถค้นหาเรคคอร์ดที่มีค่าคีย์เป็น #ID ในไฟล์ EMPOYE ซึ่งในบรรทัดถัดมารหัสดำเนินการ IFEQ ใช้ในการตรวจสอบ Indicator 91 ในสถานะ OFF ซึ่งจะถูกแปลความหมายตามตารางที่ 4

3.3.2.5 การตรวจสอบการควบคุม (Detect control)

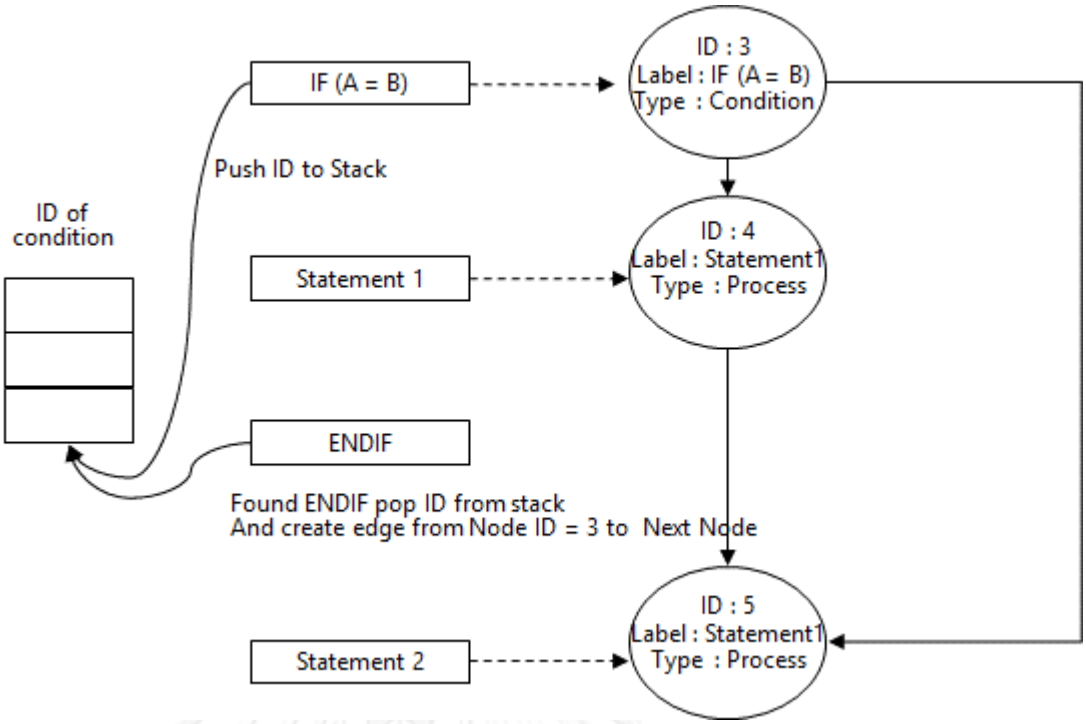
ในสวนของการตรวจสอบการควบคุมจะเป็นการตรวจสอบรหัสดำเนินการที่เป็นที่เป็น รหัสดำเนินการควบคุม เพื่อทำการควบคุมการสร้างโหนด และเส้นเชื่อมระหว่างโหนด ซึ่งการควบคุมในโปรแกรมสามารถแบ่งได้เป็น

- รหัสดำเนินการที่มีการทำงานแบบเป็นลำดับ

ในส่วนรหัสดำเนินการที่มีการทำงานแบบเป็นลำดับ จะทำการสร้างโหนดใหม่ จากนั้นสร้างเส้นเชื่อมจากโหนดสุดท้ายกับโหนดใหม่ และกำหนดโหนดใหม่เป็นโหนดสุดท้ายเพื่อใช้ในการเชื่อมกับโหนดถัดไป

- รหัสดำเนินการที่มีการทำงานแบบแยกตามเงื่อนไข

ในส่วนรหัสดำเนินการที่มีการทำงานแบบแยกตามเงื่อนไข จะทำการเก็บโหนดเงื่อนไขไว้ในสแต็ก)Stack (โดยเมื่อพบรหัสดำเนินการที่เป็นจุดจบของเงื่อนไขจะทำการนำโหนดที่เป็นเงื่อนไขออกจากสแต็ก และสร้างเส้นเชื่อมกับโหนดถัดไป ดังภาพที่ 10



ภาพที่ 10 การสร้างกราฟของรหัสดำเนินการที่มีการทำงานแบบแยกเป็นเงื่อนไข (IF)

- รหัสดำเนินการที่มีการทำงานแบบวนซ้ำตามเงื่อนไข

ในส่วนขอรหัสดำเนินการที่มีการทำงานแบบวนซ้ำตามเงื่อนไข จะมีขั้นตอนการสร้างกราฟแบบมีทิศทางเหมือนกับรหัสดำเนินการที่มีการทำงานแบบเป็นเงื่อนไข

3.3.3 การแปลงกราฟแบบมีทิศทางให้อยู่ในรูปแบบภาษากำกับเพิ่มดอท

ภาษากำกับเพิ่มดอทเป็นภาษาคำอธิบายกราฟ (Graph description language) ซึ่งเป็นข้อความที่เรียบง่ายใช้ในการอธิบายแผนภาพกราฟ และสามารถนำเข้าโปรแกรมจินตทัศน์กราฟวิซ (GraphViz) เพื่อสร้างเป็นแผนภาพกราฟแบบมีทิศทาง4][[5] โดยภาษากำกับเพิ่มดอทมีไวยากรณ์ดังภาพที่ 11

```
    graph : [ strict ] (graph | digraph) [ ID ] '{' stmt_list '}'
stmt_list : [ stmt [ ';' ] [ stmt_list ] ]
     stmt : node_stmt
          |edge_stmt
          |attr_stmt
          |ID '=' ID
          |subgraph
attr_stmt : (graph | node | edge) attr_list
attr_list : '[' [ a_list ] ']' [ attr_list ]
   a_list : ID '=' ID [ (';' | ',') ] [ a_list ]
edge_stmt : (node_id | subgraph) edgeRHS [ attr_list ]
  edgeRHS : edgeop (node_id | subgraph)[ edgeRHS ]
```

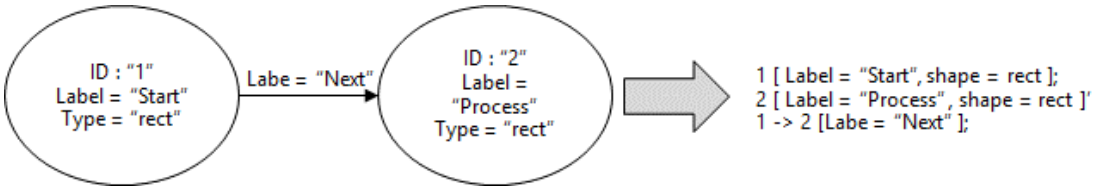
ภาพที่ 11 ไวยากรณ์ภาษากำกับเพิ่มดอท4][

จากภาพที่ 11 เป็นไวยากรณ์ภาษากำกับเพิ่มดอท ข้อความในวงเล็บ ( ) ระบุกลุ่มที่จำเป็นต้องกำหนด ข้อความในวงเล็บใหญ่ [ ] ระบุกลุ่มที่เป็นตัวเลือกสามารถเลือกใส่ หรือไม่ใส่ ตัวอักษรหนา ระบุ Terminals ตัวอักษรเอียงระบุ Nonterminals และ ข้อความที่อยู่ในอัญประกาศ (Single Quotes) ‘’ ระบุตัวอักษรตามตัวอักษร (Literal Characters)

รูปแบบการแปลงโหนดของกราฟแบบมีทิศทางเป็นภาษากำกับเพิ่มดอทแสดงในภาพที่ 12 โดยที่บรรทัดที่ 1 เป็นการระบุคุณสมบัติของโหนด ได้แก่ Node.ID คือ ไอดีของโหนดของกราฟแบบมีทิศทาง Node.Labe คือ ข้อความที่ได้จากการแปลงซอสโค้ดภาษาอาร์พีจีในหนึ่งโหนดของกราฟแบบมีทิศทาง และ Node.Type คือ ประเภทของโหนด บรรทัดที่ 2 เป็นการระบุคุณสมบัติของเส้นเชื่อมระหว่างโหนด ได้แก่ SourceNode.ID คือ โหนดต้นทาง Destination.Node.ID คือ โหนดปลายทาง และ Edge.Label คือ ข้อความกำกับเส้นเชื่อมระหว่างโหนด

```
1   Node.ID [ Label = "Node.Label" , shape = Node.Type ]
2   SourceNode.ID -> DestinationNode.ID [ labe = "Edge.Label" ]
```

ภาพที่ 12 รูปแบบการแปลง โหนดและเส้นเชื่อมของกราฟแบบมีทิศทางเป็นภาษากำกับเพิ่มดอท



ภาพที่ 13 ตัวอย่างการแปลงโหนดของกราฟแบบมีทิศทางเป็นภาษากำกับเพิ่มดอท

ภาพที่ 13 เป็นตัวอย่างการแปลงโหนดของกราฟแบบมีทิศทางเป็นภาษากำกับเพิ่มดอท ซึ่งประกอบด้วยโหนดของกราฟแบบมีทิศทาง 2 โหนด มีโหนดไอดีเป็น 1 และ 2 มีข้อความของแต่ละโหนดเป็น Start และ Procss และมีข้อความระหว่างเส้นเชื่อมเป็น Next

3.4 โปรแกรมจินตทัศน์ GraphViz [4]

ในการพัฒนาตัวแสดงผลเป็นแผนภาพของงานวิจัยนี้ได้เลือกใช้เครื่องมือกราฟวิช เข้ามาช่วยในการวาดแผนภาพผังงาน ซึ่งโปรแกรมจินตทัศน์กราฟวิช เป็นโปรแกรมโอเพนซอร์สที่แสดงโครงสร้างของข้อมูลเป็นแผนภาพกราฟ และเครือข่าย โดยโปรแกรมจะนำเข้ารายละเอียดของแผนภาพกราฟในรูปแบบของภาษาที่เรียบง่าย (Simple text language) และสามารถสร้างแผนภาพได้หลายรูปแบบ เช่น สามารถสร้างกราฟในรูปแบบของรูปภาพ สามารถสร้างกราฟในรูปแบบของ PDF หรือ โพสสคริปหรือ สามารถแสดงกราฟผ่านเบราว์เซอร์เชิงโต้ตอบ (Interactive Graph Browser) เป็นต้น 4][

บทที่ 4

การออกแบบและพัฒนาระบบ

4.1 สถาปัตยกรรมระบบ

ระบบที่พัฒนาขึ้นแบ่งออกเป็น 3 ส่วนใหญ่ คือ 1) ส่วนการสร้างกราฟแบบมีทิศทางจากซอสโค้ด 2) ส่วนแปลงกราฟมีทิศทางเป็นภาษากำกับเพิ่มดอท 3) ส่วนต่อประสานผู้ใช้งาน โดยระบบจะเริ่มต้นด้วยการนำเข้าข้อมูลนำเข้าคือไฟล์ .text ของซอสโค้ดภาษาอาร์พีจี มาผ่านกระบวนการแปลงซอสโค้ดให้อยู่ในรูปของกราฟแบบมีทิศทาง จากนั้นนำกราฟแบบมีทิศทางที่ได้แปลงเป็นภาษากำกับเพิ่มดอท .DOT เพื่อนำเข้าโปรแกรมจิตทัศน์ ได้ผลลัพธ์เป็นรูปภาพผังงานแสดงขั้นตอนการทำงานของซอสโค้ดอาร์พีจี

4.2 สภาพแวดล้อมและเครื่องมือที่ใช้ในการพัฒนา

สภาพแวดล้อมที่ใช้ในการพัฒนาระบบประกอบด้วยรายการฮาร์ดแวร์และซอฟต์แวร์ดังต่อไปนี้

4.2.1 สภาพแวดล้อม

1. หน่วยประมวลผลอินเทล คอร์ ไอ5-1.70กิกะเฮิร์ต (CPU Intel Core i53317U 1.70GHz)
2. หน่วยความจำ 8 กิกะไบต์ (8 GB RAM)
3. ฮาร์ดดิสก์ความจุ 250 กิกะไบต์ (250 GB HDD)
4. ระบบปฏิบัติการไมโครซอฟต์วินโดว์ 8.1 (Microsoft Windows 8.1) แบบ 64 บิต

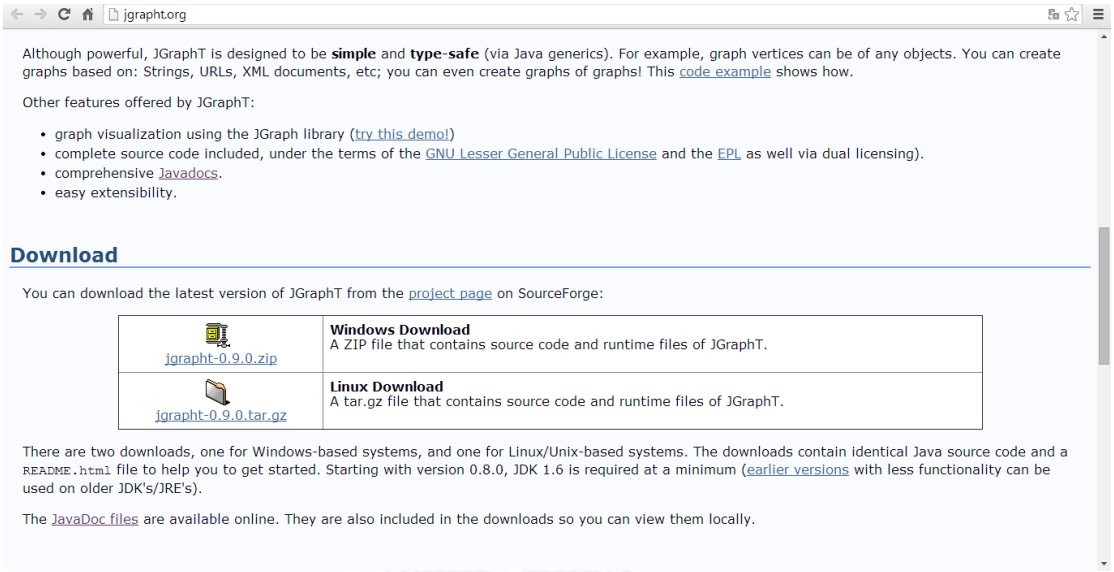
4.2.2 เครื่องมือที่ใช้ในการพัฒนา

1. อีคลิปส์ เครปเลอร์ (Eclips Kepler)

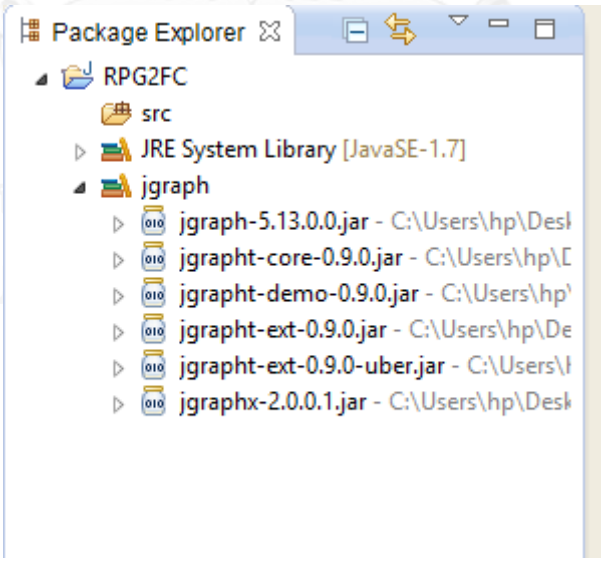
4.3 การพัฒนาระบบ

4.3.1 การพัฒนาส่วนสร้างกราฟแบบมีทิศทางจากซอสโค้ดอาร์พีจี

การพัฒนาส่วนสร้างกราฟแบบมีทิศทางจะใช้ไลบรารี (Library) JgraphT[6] ซึ่งเป็นฟรีจาวากราฟไลบรารี เป็นพื้นฐานในการสร้างกราฟแบบมีทิศทาง ซึ่งสามารถดาวโหลดได้จาก www.jgrapht.org ดังภาพที่ 14 จากนั้นทำการเพิ่มไลบรารีในโปรเจคดังภาพที่ 15



ภาพที่ 14 www.jgrapht.org



ภาพที่ 15 ไลบรารี JgraphT ที่เพิ่มเข้ามาในโปรเจค

4.3.1.1 คลาสกำหนดคุณลักษณะของกราฟแบบมีทิศทาง

การพัฒนาส่วนการสร้างกราฟแบบมีทิศทางจะเริ่มต้นจากการสร้างคลาสเพื่อกำหนดคุณลักษณะในแต่ละโหนดของกราฟแบบมีทิศทาง และการสร้างคลาสเพื่อกำหนดคุณลักษณะของเส้นเชื่อมระหว่างโหนดของกราฟแบบมีทิศทาง ได้แก่คลาส VertexNode ดังภาพที่ 15 และคลาส RelationshipEdge ดังภาพที่ 16

```
import org.jgrapht.graph.DefaultEdge;

public class VertexNode {
	int id ;
	String type = new String();
	String text = new String();
	public  VertexNode(int ID, String Type, String Text)
	{
		this.id   = ID;
		this.type = Type;
		this.text = Text;
	}
	public int getId()
	{
		return id;
	}
	public void setId(int id)
	{
		this.id = id;
	}
	public String getType()
	{
		return type;
	}
	public void setType(String type)
	{
		this.type = type;
	}
	public String getText() {
		return text;
	}
	public void setText(String text) {
		this.text = text;
	}
```

ภาพที่ 16 คลาสคุณลักษณะโหนดของกราฟแบบมีทิศทาง

จากภาพที่ 16 เป็นภาพคลาสที่ใช้เก็บข้อมูลในแต่ละโหนดของกราฟแบบมีทิศทางซึ่งประกอบด้วย id, type และ text เป็นคุณลักษณะ โดยที่ id เป็นตัวแทนของแต่ละโหนด type เป็นตัวระบุประเภทของโหนด (rect, dimond) และ text เป็นตัวระบุข้อความของแต่ละโหนด

```
public static class RelationshipEdge<VertextNode> extends DefaultEdge {
	private VertexNode Vertex1;
	private VertexNode Vertex2;
	private String label;
	public RelationshipEdge(VertexNode v1, VertexNode v2, String label)
	{
		this.Vertex1 = v1;		//Source Node
		this.Vertex2 = v2;		//Destination Node
		this.label = label;
	}
	public VertexNode getVertex1()
	{
		return Vertex1;
	}
	public void setVertex1(VertexNode vertex1)
	{
		Vertex1 = vertex1;
	}
	public VertexNode getVertex2()
	{
		return Vertex2;
	}
	public void setVertex2(VertexNode vertex2)
	{
		Vertex2 = vertex2;
	}
	public String getLabel()
	{
		return label;
	}
	public void setLabel(String label)
	{
		this.label = label;
	}
}
```

ภาพที่ 17 คลาสเส้นเชื่อมระหว่างโหนดของกราฟแบบมีทิศทางที่สืบทอดจากคลาส DefaultEdge ของไลบรารี่ JgraphT

จากภาพที่ 17 เป็นคลาสความสัมพันธ์ หรือเส้นเชื่อมของกราฟแบบมีทิศทางที่สืบทอดจากคลาส DefaultEdge ของไลบรารี Jgrapht โดยมีการเพิ่มคุณลักษณะ label เพื่อเป็นข้อความกำกับเส้นเชื่อมระหว่างโหนด

4.3.1.2 ขั้นตอนการสร้างกราฟแบบมีทิศทาง

ขั้นตอนการสร้างกราฟแบบมีทิศทางเริ่มต้นจากการอ่านซอสโค้ดภาษาอาร์พีจีที่อยู่ในรูปเท็กซ์ไฟล์ทีละบรรทัด จากนั้นตรวจสอบซอสโค้ดตำแหน่งที่ 6 และตำแหน่งที่ 7 หากซอสโค้ดตำแหน่งที่ 6 มีค่าเป็น C ซึ่งหมายถึงซอสโค้ดบรรทัดนั้นเป็น Calculation Specification และ ซอสโค้ดตำแหน่งที่ 7 ไม่ใช่ * ซึ่งหมายถึงซอสโค้ดบรรทัดนั้นไม่ใช่คอมเม้นของโปรแกรม จะทำการตัดแบ่งซอสโค้ดออกเป็น 6 ส่วน ได้แก่

- Factor 1 ซอสโค้ดตำแหน่งที่ 18-27
- Operation code ซอสโค้ดตำแหน่งที่ 28-32
- Factor 2 ซอสโค้ดตำแหน่งที่ 33-42
- Result Field ซอสโค้ดตำแหน่งที่ 43-48
- Indicator 1 ซอสโค้ดตำแหน่งที่ 54-55
- Indicator 2 ซอสโค้ดตำแหน่งที่ 56-57
- Indicator 3 ซอสโค้ดตำแหน่งที่ 58-59

สำหรับทุกๆรหัสดำเนินการจะมีคลาสรองรับเพื่อสร้างโหนด และความสัมพันธ์ระหว่างโหนด โดยเมื่อโปรแกรมตรวจสอบพบรหัสดำเนินการใดๆ โปรแกรมจะทำการเรียกคลาสของรหัสดำเนินการนั้นเพื่อทำการสร้างโหนด และเส้นเชื่อมของกราฟแบบมีทิศทาง ซึ่งแต่ละคลาสของรหัสดำเนินการจะมีการแปลงข้อความของโหนดตามตาราง ก ในภาคผนวก ก

คลาสของรหัสดำเนินการทุกรหัสดำเนินการจะสืบทอดมาจากจากคลาส RPGElement และมีชื่อคลาสเป็น RPGopt ต่อด้วยรหัสดำเนินการ เช่น คลาสของรหัสดำเนินการ ADD จะมีชื่อคลาสเป็น RPGoptADD เป็นต้น

4.3.1.3 คลาส RPGElement

ในส่วนของคลาส RPGElement ดังภาพที่ 18 จะประกอบไปด้วยข้อมูลของซอสโค้ด ได้แก่ รหัสดำเนินการ, Factor 1, Factor 2, Result field, Resulting Indicator ซึ่งข้อมูลของซอสโค้ดจะถูกนำเข้าด้วย Constructor และคลาส RPGElement จะมีข้อมูลนำออก คือ โหนด และ เส้นเชื่อมของกราฟแบบมีทิศทางที่ได้จากการแปลงรหัสดำเนินการ

สำหรับคลาสของแต่ละโอเปอเรชันโค้ดที่สืบทอดจากคลาส RPGElement จะมีการทำงานในส่วนเมท็อด process () ที่แตกต่างกัน ซึ่งในส่วนเมท็อด process () จะเป็นส่วนที่ใช้ในการสร้างโหนด เส้นเชื่อม คำอธิบายโหนด คำอธิบายเส้นเชื่อม ของกราฟแบบมีทิศทาง ตัวอย่างคลาสของรหัสดำเนินการ ADD แสดงดังภาพที่ 19 ตัวอย่างคลาสของรหัสดำเนินการ CHAIN แสดงดังภาพที่ 20 และตัวอย่างคลาสของรหัสดำเนินการ IFEQ แสดงดังภาพที่ 21

```
public class RPGElement {
	String operation = new String();
	String factor1 = new String();
	String factor2 = new String();
	String factor3 = new String();
	String indicator1 = new String();
	String indicator2 = new String();
	String indicator3 = new String();
	ArrayList<VertexNode> node = new ArrayList<VertexNode>();
	ArrayList<RelationshipEdge> edge = new
ArrayList<RelationshipEdge>();

	public void setConstructor(String opt, String f1, String f2, String
rst,String in1, String in2, String in3)
	{	this.operation = opt;
		this.factor1 = f1;
		this.factor2 = f2;
		this.factor3 = rst;
		this.indicator1 = in1;
		this.indicator2 = in2;
		this.indicator3 = in3;
	}
	public void process()
	public ArrayList<RelationshipEdge> getEdge() {
		return edge;
	}
	public ArrayList<VertexNode> getNode() {
		return node;
	}
}
```

ภาพที่ 18 คลาส RPGElement

```
public class RPGoptADD extends RPGElement{
	String text = new String();
	public void process()	{
		//Set text
		if (factor1.compareTo("")==1)
		{
			// text: result = factor1 + factor2
			text = factor3 + " = " + factor1 + " + " + factor2;
		}
		else
		{
			// text: result = result + factor2
			text = factor3 + " = " + factor3 + " + " + factor2;
		}
		//Get and update ID
		int idVertex = GenMetadata.getId()+1;
		GenMetadata.setId(idVertex);

		//Crate new vertex
		VertexNode newVertex = new VertexNode(idVertex,"rect",text);

		//Add vertex
		node.add(newVertex) ;

		//Add edge
		edge.add(
		new RelationshipEdge(GenMetadata.lastVertex,newVertex,"")) ;

		//Set lastVertex
		GenMetadata.setLastVertex(newVertex);
	}
}
```

ภาพที่ 19 ตัวอย่างคลาสที่สืบทอดจากคลาส RPGElement ของรหัสดำเนินการ ADD

ภาพที่ 19 เป็นคลาสที่สืบทอดจากคลาส RPGElement เมื่อมีการนำเข้ารหัสดำเนินการ ADD โปรแกรมจะทำการเรียนกคลาส RPGoptADD เพื่อทำการสร้างโหนด และเส้นเชื่อมในเมท็อด process ()

```
public class RPGoptCHAIN extends RPGElement {
   String text = new String();
   public void process()
   {
     //Set Text
     text = "Search file " + factor2 + " By key " + factor1;

     //Get and update ID
     int idVertex = GenMetadata.getId()+1;
     GenMetadata.setId(idVertex);

     //Crate new vertex
     VertexNode newVertex = new VertexNode(idVertex,"rect",text);

     //Add vertex
     node.add(newVertex) ;

     //Add edge
     edge.add(new RelationshipEdge(GenMetadata.lastVertex,newVertex,""));

     //Set lastVertex
     GenMetadata.setLastVertex(newVertex);

     //Indicator setup
     IndicatorElement indicatorElement = new IndicatorElement();
     indicatorElement.setIndocator(indicator1);
     //Set Text when status of indicator1 is ON
     indicatorElement.setoffIndocator("Search file "+ factor2 + " by key
" + factor1 + " not found" );
     //Set Text when status of indicator1 is OFF
     indicatorElement.setoffIndocator("Search file "+ factor2 + " by key
" + factor1 + " found" );

     //Add Indicator element to Indicator control
      GenMetadata.IndicatorControl.AddIndicator(indicatorElement);
   }
}
```

ภาพที่ 20 คลาสที่สืบทอดจากคลาส RPGElement ของรหัสดำเนินการ CHAIN

ภาพที่ 20 เป็นคลาสที่สืบทอดจากคลาส RPGElement เมื่อมีการนำเข้ารหัสดำเนินการ CHAIN โปรแกรมจะทำการเรียกคลาส RPGoptCHAIN เพื่อทำการสร้างโหนด และเส้นเชื่อม รวมทั้งเก็บค่า Indicator1 โดยที่หาก Indicator1 มีสถานะเป็น ON หมายถึง ไม่สามารถค้นหาไฟล์เจอ เมื่อ Indicator2 มีสถานะเป็น OFF หมายถึง สามารถค้นหาไฟล์เจอ ซึ่งค่าของ Indicator จะถูกเก็บไว้ใน Indicator control เพื่อนำไปประกอบการแปลความหมายของข้อความรหัสดำเนินการถัดไป

```
public class RPGoptIFEQ extends RPGElement
{
   String text = new String();
   public void process()
   {
       // Case check indicator
       if (factor1.substring(0,3).equals("*IN"))
       {
          //Status of indicator is OFF
          if (factor2.equals("*OFF"))
          {
             IndicatorElement indicatorElement = new IndicatorElement();
             indicatorElement =
GenMetadata.IndicatorControl.SearchIndicatro(factor1.substring(3,5));
              text = indicatorElement.getoffIndicator();
          }
          //Status of indicator is ON
          else if (factor2.equals("*ON"))
          {
             IndicatorElement indicatorElement = new IndicatorElement();
             indicatorElement =
GenMetadata.IndicatorControl.SearchIndicatro(factor1.substring(3,5));
             text = indicatorElement.getonIndicator();
          }
       }
       else if(factor2.substring(0,3).equals("*IN"))
       {
          //Status of indicator is OFF
          if (factor1.equals("*OFF"))
          {
              IndicatorElement indicatorElement = new IndicatorElement();
              indicatorElement =
GenMetadata.IndicatorControl.SearchIndicatro(factor1.substring(3,5));
              text = indicatorElement.getoffIndicator();
          }
          //Status of indicator is ON
          else if (factor1.equals("*ON"))
          {
              IndicatorElement indicatorElement = new IndicatorElement();
              indicatorElement =
GenMetadata.IndicatorControl.SearchIndicatro(factor1.substring(3,5));
              text = indicatorElement.getonIndicator();
          }
       }
       // Not concern with indicator status
       else
       {
          text = " ( " + factor1 + " = " + factor2 + " )";
       }
```

```
      //Get and update ID
      int idVertex = GenMetadata.getId()+1;
      GenMetadata.setId(idVertex);

      //Crate new vertex
      VertexNode newVertex = new VertexNode(idVertex,"diamond",text);

      //Add vertex
      node.add(newVertex) ;

      //Add edge
      edge.add(new RelationshipEdge(GenMetadata.lastVertex,newVertex,""));

      //Set lastVertex
      GenMetadata.setLastVertex(newVertex);

      //Add stack
      GenMetadata.stack.push(newVertex);
      }
}
```

ภาพที่ 21 คลาสที่สืบทอดจากคลาส RPGElement ของรหัสกำเนินการ IFEQ

4.3.1.4 การเก็บข้อมูล Resulting Indicator และการสืบค้น Resulting Indicator

การพัฒนาในส่วนการควบคุม Indicator ในโปรแกรม จะประกอบด้วยคลาส IndicatorElement และ คลาส IndicatorControl ดังภาพที่ 22 และ ภาพที่ 23

```
public class IndicatorElement {

	//Indicator No.
	String Indicator = new String();

	//String when indicator on
	String onIndicator = new String();

	//String when indicator off
	String offIndicator = new String();

	public void setIndocator(String a)
	{
		Indicator = a;
	}
	public void setonIndocator(String a)
	{
		onIndicator = a;
	}
	public void setoffIndocator(String a)
	{
		offIndicator = a;
	}
	public String getIndicator ()
	{
		return Indicator;
	}
	public String getonIndicator ()
	{
		return onIndicator;
	}
	public String getoffIndicator ()
	{
		return offIndicator;
	}
}
```

ภาพที่ 22 เป็นคลาส IndicatorElement

ภาพที่ 22 คลาส IndicatorElement ซึ่งจะถูกสร้างเป็นอ็อบเจกต์ (Object) เมื่อโปรแกรมนำเข้า รหัสดำเนินการที่มีเอ้าท์พุทในส่วนของ Resulting Indicator โดยคุณลักษณะของคลาส IndicatorElement จะประกอบด้วย

- Indicator คือ ตัวแทน Indicator หรือ ชื่อ Indicator
- onIndicator คือ ข้อความเมื่อ Indicator อยู่ในสถานะ ON
- offIndicator คือ ข้อความเมื่อ Indicator อยู่ในสถานะ OFF

```
public class IndicatorControl {

   //Array for keep indicator element
   public static ArrayList<IndicatorElement> stackIndicator = new
ArrayList<IndicatorElement>();

   public void AddIndicator(IndicatorElement a)
   {
       stackIndicator.add(a);
   }

   public IndicatorElement SearchIndicatro(String SearchIndicator)
   {
       int i ;
       for (i = stackIndicator.size()-1; i <= 0 ; i--)
       {
         if(stackIndicator.get(i).getIndicator().equals(SearchIndicator))
              {break;}
       }
       return stackIndicator.get(i);
   }
}
```

ภาพที่ 23 คลาส IndicatorControl

ภาพที่ 23 เป็นคลาส IndicatorControl ซึ่งเป็นคลาสที่ใช้สำหรับควบคุม Indicator ทั้งหมดในโปรแกรม โดยจะเก็บอ็อบเจกต์ IndicatorElement ในรายการของแถวลำดับ (Array list) ด้วยเมท็อด AddIndicator และ สามารถสืบค้น Indicator รวมทั้งข้อความเมื่อ Indicator นั้นมีสถานะเป็น ON หรือ OF ด้วยเมท็อด SearchIndicator

4.3.2 การพัฒนาตัวแปลงกราฟแบบมีทิศทางเป็นภาษากำกับเพิ่มดอท

การพัฒนาตัวแปลงกราฟแบบมีทิศทางเป็นภาษากำกับเพิ่มดอท เป็นการแปลงกราฟแบบมีทิศทางที่อยู่ในรูปของ JgrahT ให้เป็นภาษากำกับเพิ่มดอท ซึ่งกราฟแบบมีทิศทางในรูปแบบภาษากำกับเพิ่มดอทนั้น ประกอบด้วย 2 ส่วนหลักๆ ได้แก่ ส่วนของการกำหนดคุณสมบัติของโหนด และส่วนของการกำหนดความสัมพันธ์ หรือเส้นเชื่อมระหว่างโหนด ดังภาพที่ 24 บรรทัดที่ 1-4 เป็นการกำหนดคุณสมบัติของโหนด และบรรทัดที่ 5-6 เป็นการกำหนดความสัมพันธ์ระหว่างโหนด

```
1 digraph G {
2 A [label = “label A”, shape = “ ” ] ;
3 B [label = “label B”, shape = “ ” ] ;
4 C [label = “label C”, shape = “ “ ] ;
5 A->B ;
6 B->C ;
7 }
```

ภาพที่ 24 ตัวอย่างภาษากำกับเพิ่มดอท

คลาสสำหรับการแปลงกราฟแบบมีทิศทางเป็นภาษากำกับเพิ่มดอทในภาพที่ 25 จะทำการสร้างภาษากำกับเพิ่มดอทในส่วนของการกำหนดคุณสมบัติของโหนด ด้วยการเรียกใช้เมท็อด vertexSet() ซึ่งจะส่งค่ากลับคืนเป็นเซต (Set) ของโหนดทั้งหมดในกราฟแบบมีทิศทาง จากนั้นทำการแปลงคุณลักษณะของโหนดทั้งหมดที่ได้จากเมท็อด vertexSet() เป็นภาษาดอท

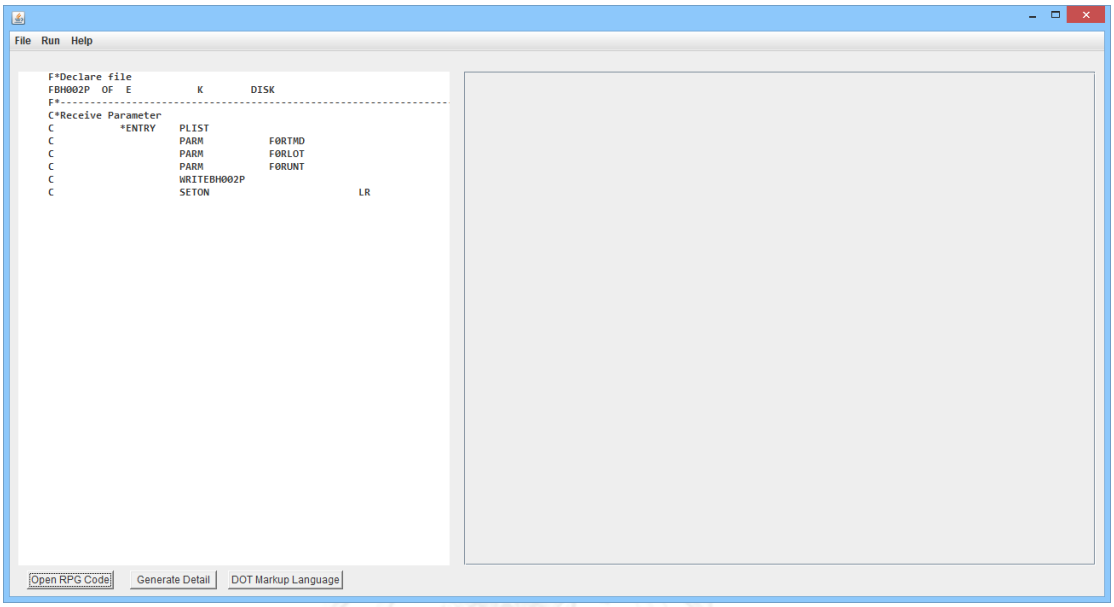
ในส่วนของการกำหนดความสัมพันธ์ระหว่างโหนดจะทำการเรียกใช้เมท็อด outgoingEdgesOf (vertex) ซึ่งจะส่งค่ากลับคืนเป็นเซตของเส้นเชื่อมทั้งหมดของโหนดที่ระบุ เพื่อทำการดึงคุณลักษณะของเส้นเชื่อมจากกราฟแบบมีทิศทางมาสร้างเป็นภาษากำกับเพิ่มดอท

```
public class GenFileDOT {
public void genarateDOT(DirectedGraph<VertexNode, RelationshipEdge>
metadatagraph)
{
	BufferedWriter writer = null;
	try {
	String FileName = "DotFile.dot";
	File DotFile = new File(FileName);
	DotFile.delete();

	//Start command
	writer.write("digraph G {");
	writer.newLine();

	//Generate Node Attribute
	for (VertexNode vertex : metadatagraph.vertexSet())
	{
		writer.write (  vertex.getId()
					+"[ label = \""
					+ vertex.getText()
					+"\""
					+ " ,shape = "
					+ vertex.getType()
					+" ] ;"  +"\n");
	}

	//Generate Edge Attribute
	for (VertexNode vertex : metadatagraph.vertexSet()) {
	for (RelationshipEdge edge : metadatagraph.outgoingEdgesOf(vertex))
	{
		writer.write ( edge.getVertex1().getId()
				+ " -> "
				+ edge.getVertex2().getId()
				+ "[label = \" "
				+ edge.getLabel()
				+" \" ] ;" + "\n");
	}
	}

	//End command
	writer.write("}");
	writer.newLine();
	}
	catch (Exception e)
	{
		e.printStackTrace();
	}
	finally
	{	try
		{writer.close();    }
		catch(Exception e){}
	}
}
```

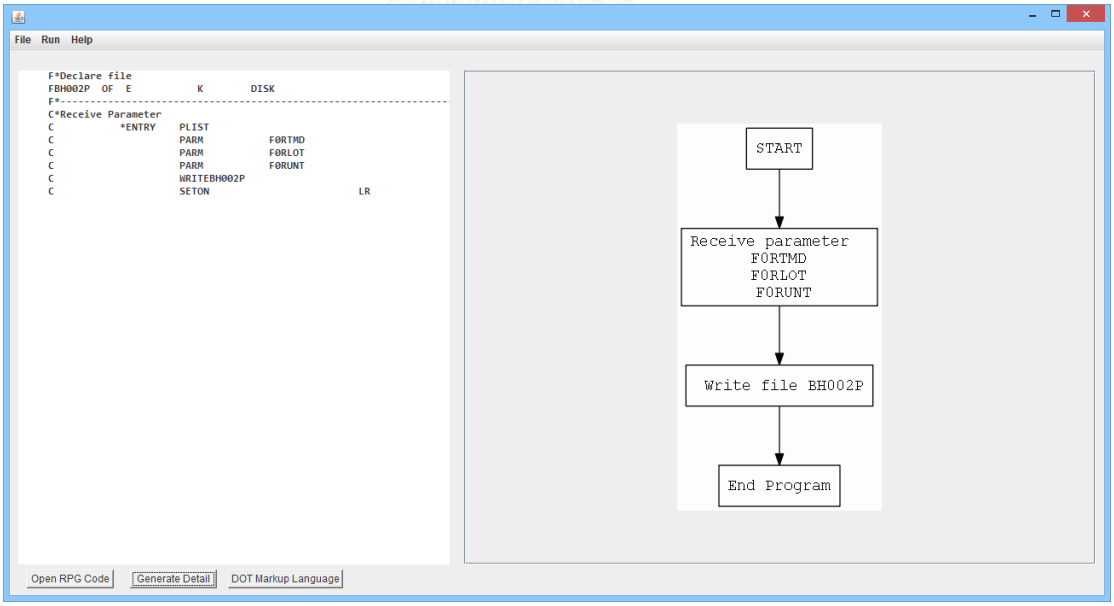
ภาพที่ 25 คลาสสำหรับการแปลงกราฟแบบมีทิศทางเป็นภาษากำกับเพิ่มดอท

4.3.3 การพัฒนาส่วนแสดงผล

ส่วนแสดงผลจะแบ่งออกเป็น 2 ส่วน ได้แก่ ส่วนแสดงซอสโค้ดภาษาอาร์พีจี และส่วนแสดงแผนภาพผังงาน ดังภาพที่ 26 และ ภาพที่ 27



ภาพที่ 26 ส่วนแสดงผลโปรแกรมแปลงภาษาอาร์พีจีเป็นผังงาน



ภาพที่ 27 ส่วนแสดงผลโปรแกรมแปลงภาษาอาร์พีจีเป็นผังงาน เมื่อเปิดซอสโค้ดภาษาอาร์พีจี และแสดงภาพผังงาน

บทที่ 5

การประเมินและการวัดผล

5.1 แนวทางการประเมินผลงานวิจัย

แนวทางการประเมินจะใช้การเปรียบเทียบระหว่างซอสโค้ดภาษาอาร์พีจี และแผนภาพผังงานที่ได้จากการแปลงซอสโค้ด โดยข้อมูลตัวอย่างจะแบ่งออกเป็น 3 กลุ่ม คือ กลุ่มซอสโค้ดที่มีการทำงานเป็นลำดับ กลุ่มซอสโค้ดที่ทำงานแบบแยกตามเงื่อนไข และกลุ่มซอสโค้ดที่ทำงานแบบวนซ้ำตามเงื่อนไข และซอสโค้ดที่เกี่ยวข้องกับ Resulting indicator โดยแนวทางการประเมินประกอบด้วยรายละเอียดดังต่อไปนี้

1. ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี
2. ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ
3. ผลลัพธ์รูปภาพผังงาน
    - ตรวจสอบลำดับของกระบวนการแต่ละกระบวนการในผังงาน
    - ตรวจสอบคำอธิบายของแต่ละกระบวนการในผังงาน
    - ตรวจสอบสัญลักษณ์ของผังงาน

5.2 ผลการเปรียบเทียบตัวอย่างที่ 1

ตัวอย่างที่ 1 เป็นตัวอย่างของการแปลงของซอสโค้ดที่มีการทำงานเป็นลำดับ แสดงให้เห็นถึงความสามารถในการแปลงแผนภาพผังงานได้ถูกต้องตามลำดับการทำงานในซอสโค้ด และมีลักษณะของผังงานรวมทั้งคำอธิบายที่ถูกต้องตามซอสโค้ด

5.2.1 ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี

ข้อมูลนำเข้าซอสโค้ดอาร์พีจีตัวอย่างที่ 1 ดังภาพที่ 28 ประกอบด้วยรหัสดำเนินการ PLIST, PARM และ WRITE ซึ่งทั้ง 3 รหัสดำเนินการเป็นรหัสดำเนินการที่มีการทำงานเป็นลำดับ

```
File  Edit  Format  View  Help
     F*Declare file
     FBH002P  OF  E           K        DISK
     F*-------------------------------------------------------------------------
     C*Receive Parameter
     C           *ENTRY    PLIST
     C                     PARM           F0RTMD
     C                     PARM           F0RLOT
     C                     PARM           F0RUNT
     C*Write file BH002P
     C                     WRITEBH002P
     C*End Program
     C                     SETON                        LR
```

ภาพที่ 28 ซอสโค้ดอาร์พีจีตัวอย่างที่ 1

5.2.2 ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ

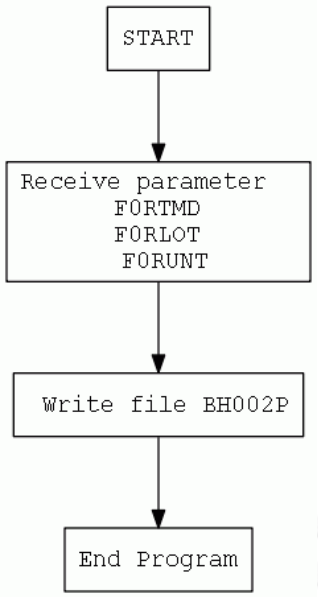
ผลลัพธ์ภาษากำกับเพิ่มดอทของซอสโค้ดตัวอย่างที่ 1 แสดงได้ดังภาพที่ 29

```
File  Edit  Format  View  Help
digraph G {
1 -> 2[label = "  " fontname = Courier ];
2 -> 3[label = "  " fontname = Courier ];
3 -> 4[label = "  " fontname = Courier ];
1[ label = "START" ,shape = Start fontname = Courier ];
2[ label = "Receive parameter  \n F0RTMD \n F0RLOT \n F0RUNT" ,shape = rect fontname = Courier ];
3[ label = " Write file BH002P" ,shape = rect fontname = Courier ];
4[ label = "End Program" ,shape = process fontname = Courier ];
{rank = sink ; 4}}
```

ภาพที่ 29 ผลลัพธ์ภาษากำกับเพิ่มดอทตัวอย่างที่ 1

5.2.3 ผลลัพธ์รูปภาพผังงาน

ผลที่ได้จากการนำเข้าภาษากำกับเพิ่มดอทตัวอย่างที่ 1 ไปสร้างแผนภาพผังงานด้วยโปรแกรมจินตทัศน์กราฟวิซเป็นรูปภาพ แสดงได้ดังภาพที่ 30



ภาพที่ 30 ผลลัพธ์รูปภาพผังงานตัวอย่างที่ 1

5.3 ผลการเปรียบเทียบตัวอย่างที่ 2

ตัวอย่างที่ 2 เป็นตัวอย่างของการแปลงของซอสโค้ดที่มีการทำงานแบบดำเนินการซ้ำตามเงื่อนไข แสดงให้เห็นถึงความสามารถในการแปลงซอสโค้ดที่มีการทำงานแบบกำเนินการซ้ำตามเงื่อนได้ถูกต้อง และมีลักษณะของผังงานรวมทั้งคำอธิบายที่ถูกต้องตามซอสโค้ด

5.3.1 ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี

ข้อมูลนำเข้าซอสโค้ดอาร์พีจีตัวอย่างที่ 2 ดังภาพที่ 31 ประกอบด้วยรหัสดำเนินการ DOWEQ ซึ่งเป็นรหัสดำเนินการที่มีการทำงานแบบกำเนินการซ้ำตามเงื่อนไข โดยจะเปรียบเทียบ Factor 1 และ Factor 2 ซึ่งจะมีการวนซ้ำเมื่อ Factor1 และ Factor2 มีค่าเท่ากัน

File Edit Format View Help

```
     FBH002P  IF  E           K        DISK
     F*-------------------------------------------------------------------------
     C           *LOVAL     SETLLBH002P
     C                      READ BH002P                   90
     C*Read file BH002P until end of file
     C           *IN90      DOWEQ*OFF
     C*Update Field update job = 'Manual'
     C                      MOVEL'MANUAL'  F0RUPJ
     C                      READ BH002P                   90
     C                      ENDDO
     C                      SETON                         LR
```

ภาพที่ 31 ซอสโค้ดอาร์พีจีตัวอย่างที่ 2

5.3.2 ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ

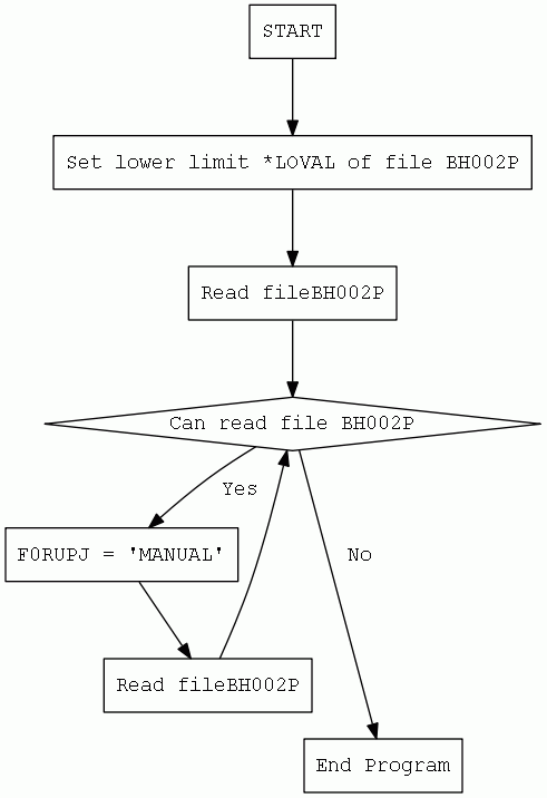
ผลลัพธ์ภาษากำกับเพิ่มดอทของซอสโค้ดตัวอย่างที่ 2 จะได้ดังภาพที่ 32

```
File  Edit  Format  View  Help
digraph G {
1 -> 2[label = "   " fontname = Courier ];
2 -> 3[label = "   " fontname = Courier ];
3 -> 4[label = "   " fontname = Courier ];
4 -> 5[label = " Yes " fontname = Courier ];
4 -> 7[label = " No " fontname = Courier ];
5 -> 6[label = "   " fontname = Courier ];
6 -> 4[label = "    " fontname = Courier ];
1[ label = "START" ,shape = Start fontname = Courier ];
2[ label = "Set lower limit *LOVAL of file BH002P" ,shape = rect fontname = Courier ];
3[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
4[ label = "Can read file BH002P" ,shape = diamond fontname = Courier ];
5[ label = "F0RUPJ = 'MANUAL'" ,shape = rect fontname = Courier ];
6[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
7[ label = "End Program" ,shape = process fontname = Courier ];
{rank = sink ; 7}}
```

ภาพที่ 32 ผลลัพธ์ภาษากำกับเพิ่มดอทตัวอย่างที่ 2

5.3.3 ผลลัพธ์รูปภาพผังงาน

ผลที่ได้จากการนำเข้าภาษากำกับเพิ่มดอทตัวอย่างที่ 2 ไปสร้างแผนภาพผังงานด้วยโปรแกรมจินตทัศน์กราฟวิซเป็นรูปภาพ แสดงได้ดังภาพที่ 33



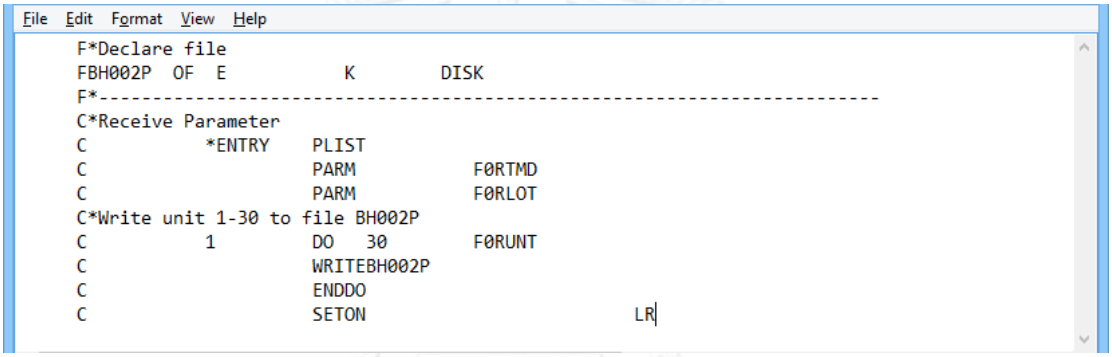
ภาพที่ 33 ผลลัพธ์รูปภาพผังงานตัวอย่างที่ 2

5.4 ผลการเปรียบเทียบตัวอย่างที่ 3

ตัวอย่างที่ 3 เป็นตัวอย่างของการแปลงของซอสโค้ดที่มีการทำงานแบบวนซ้ำ แสดงให้เห็นถึงความสามารถในการแปลงซอสโค้ดที่มีการทำงานแบบดำเนินการซ้ำตามเงื่อนไขได้ถูกต้อง และมีลักษณะของผังงานรวมทั้งคำอธิบายที่ถูกต้องตามซอสโค้ด

5.4.1 ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี

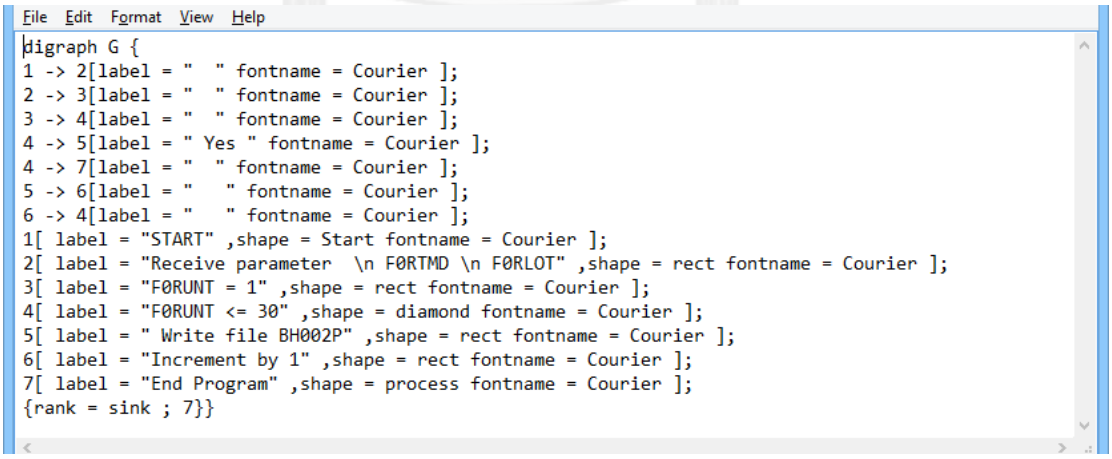
ข้อมูลนำเข้าซอสโค้ดอาร์พีจีตัวอย่างที่ 3 ดังภาพที่ 34 ประกอบด้วยรหัสดำเนินการ DO ซึ่งเป็นรหัสดำเนินการที่มีการทำงานแบบดำเนินการซ้ำตามเงื่อนไข โดยจะเปรียบเทียบ Factor 2 และ Result Field ซึ่งจะมีการวนซ้ำเมื่อ Result field มีค่าน้อยกว่า Factor 2



ภาพที่ 34 ซอสโค้ดอาร์พีจีตัวอย่างที่ 3

5.4.2 ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ

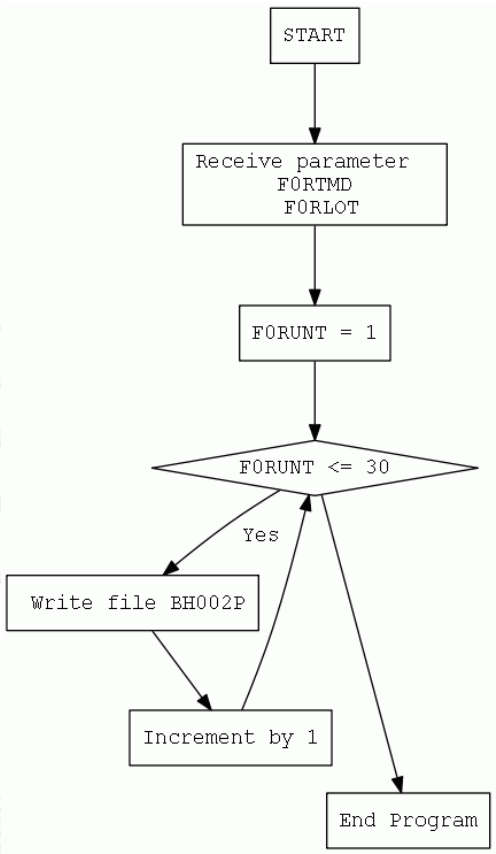
ผลลัพธ์ภาษากำกับเพิ่มดอทของซอสโค้ดตัวอย่างที่ 3 จะได้ดังภาพที่ 35



ภาพที่ 35 ผลลัพธ์ภาษากำกับเพิ่มดอทตัวอย่างที่ 3

5.4.3 ผลลัพธ์รูปภาพผังงาน

ผลที่ได้จากการนำเข้าภาษากำกับเพิ่มดอทตัวอย่างที่ 3 ไปสร้างแผนภาพผังงานด้วยโปรแกรมจินตทัศน์กราฟวิซเป็นรูปภาพ แสดงได้ดังภาพที่ 36



ภาพที่ 36 ผลลัพธ์รูปภาพผังงานตัวอย่างที่ 3

5.5 ผลการเปรียบเทียบตัวอย่างที่ 4

ตัวอย่างที่ 4 เป็นตัวอย่างของการแปลงของซอสโค้ดที่มีการทำงานแบบแยกตามเงื่อนไข แสดงให้เห็นถึงความสามารถในการแปลงซอสโค้ดที่มีการทำงานแบบแยกตามเงื่อนไขได้ถูกต้อง และมีลักษณะของผังงานรวมทั้งคำอธิบายที่ถูกต้องตามซอสโค้ด

5.5.1 ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี

ข้อมูลนำเข้าซอสโค้ดอาร์พีจีตัวอย่างที่ 5 ดังภาพที่ 37 ประกอบด้วยรหัสดำเนินการ SELEC ซึ่งเป็นรหัสดำเนินการที่มีการทำงานแบบแยกตามเงื่อนไข โดยจะมีการตรวจสอบเงื่อนไขด้วยรหัสดำเนินการ WHEQ และจบเงื่อนไขด้วยโอเปอเรชันโค้ด ENDSL

```
File  Edit  Format  View  Help
     F*Declare file
     FBH002P  OF  E           K        DISK
     F*-------------------------------------------------------------------------
     C           *ENTRY    PLIST
     C                     PARM           F0RTMD
     C                     PARM           F0RUNT
     C                     PARM           F0RCLC
     C                     SELEC
     C           F0RCLC    WHEQ '001'
     C                     MOVE 'ORANG'   F0RCLN
     C           F0RCLC    WHEQ '002'
     C                     MOVE 'BLACK'   F0RCLN
     C           F0RCLC    WHEQ '003'
     C                     MOVE 'BLUE'    F0RCLN
     C                     OTHER
     C                     MOVE 'WHITE'   F0RCLN
     C                     ENDSL
     C                     WRITEBH002P
     C                     SETON                      LR
```

ภาพที่ 37 ซอสโค้ดอาร์พีจีตัวอย่างที่ 4

5.5.2 ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ

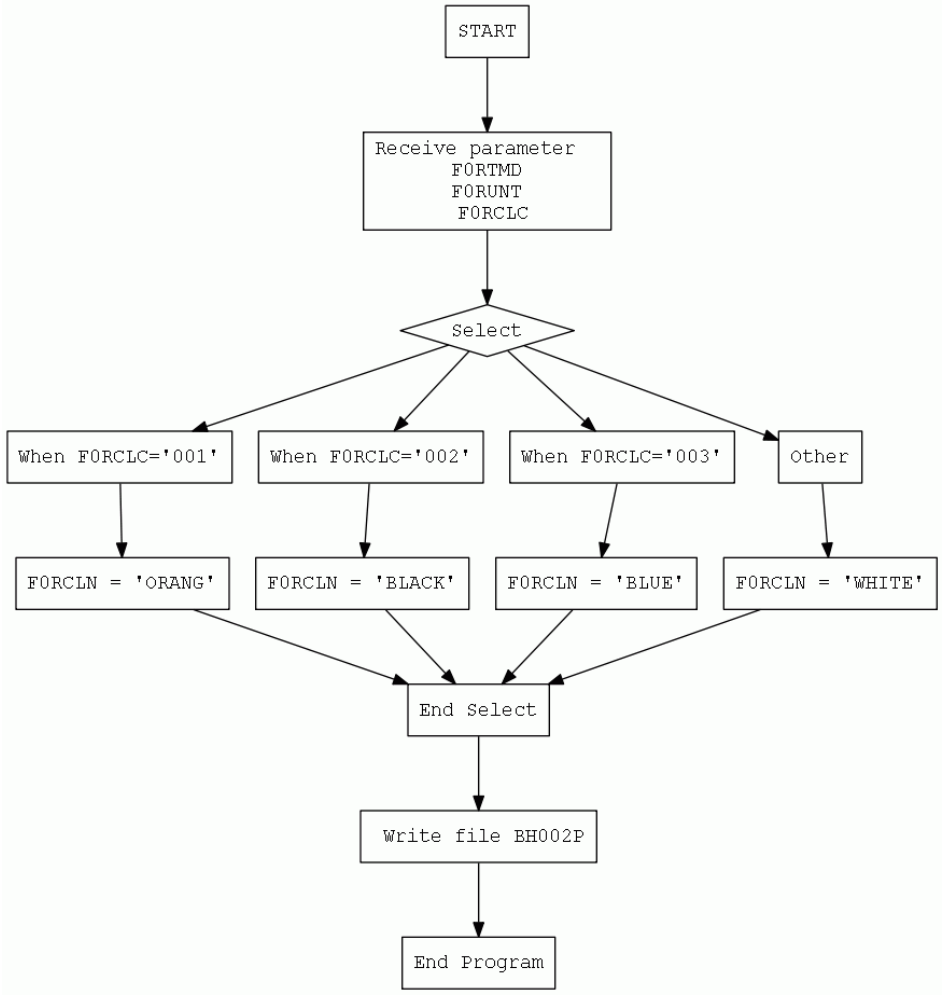
ผลลัพธ์ภาษากำกับเพิ่มดอทของซอสโค้ดตัวอย่างที่ 5 จะได้ดังภาพที่ 38

```
File  Edit  Format  View  Help
digraph G {
1 -> 2[label = "  " fontname = Courier ];
2 -> 3[label = "  " fontname = Courier ];
3 -> 5[label = "  " fontname = Courier ];
3 -> 7[label = "  " fontname = Courier ];
3 -> 9[label = "  " fontname = Courier ];
3 -> 11[label = "  " fontname = Courier ];
4 -> 13[label = "  " fontname = Courier ];
5 -> 6[label = "  " fontname = Courier ];
6 -> 4[label = "  " fontname = Courier ];
7 -> 8[label = "  " fontname = Courier ];
8 -> 4[label = "  " fontname = Courier ];
9 -> 10[label = "  " fontname = Courier ];
10 -> 4[label = "  " fontname = Courier ];
11 -> 12[label = "  " fontname = Courier ];
12 -> 4[label = "  " fontname = Courier ];
13 -> 14[label = "  " fontname = Courier ];
1[ label = "START" ,shape = Start fontname = Courier ];
2[ label = "Receive parameter  \n F0RTMD \n F0RUNT \n F0RCLC" ,shape = rect fontname = Courier ];
3[ label = "Select" ,shape = diamond fontname = Courier ];
4[ label = "End Select" ,shape = rect fontname = Courier ];
5[ label = "When F0RCLC='001'" ,shape = rect fontname = Courier ];
6[ label = "F0RCLN = 'ORANG'" ,shape = rect fontname = Courier ];
7[ label = "When F0RCLC='002'" ,shape = rect fontname = Courier ];
8[ label = "F0RCLN = 'BLACK'" ,shape = rect fontname = Courier ];
9[ label = "When F0RCLC='003'" ,shape = rect fontname = Courier ];
10[ label = "F0RCLN = 'BLUE'" ,shape = rect fontname = Courier ];
11[ label = "Other" ,shape = rect fontname = Courier ];
12[ label = "F0RCLN = 'WHITE'" ,shape = rect fontname = Courier ];
13[ label = " Write file BH002P" ,shape = rect fontname = Courier ];
14[ label = "End Program" ,shape = process fontname = Courier ];
{rank = sink ; 14}}
```

ภาพที่ 38 ผลลัพธ์ภาษากำกับเพิ่มดอทตัวอย่างที่ 4

5.5.3 ผลลัพธ์รูปภาพผังงาน

ผลที่ได้จากการนำเข้าภาษากำกับเพิ่มดอทตัวอย่างที่ 5 ไปสร้างแผนภาพผังงานด้วยโปรแกรมจินตทัศน์กราฟวิซเป็นรูปภาพ แสดงได้ดังภาพที่ 39



ภาพที่ 39 ผลลัพธ์รูปภาพผังงานตัวอย่างที่ 4

5.6 ผลการเปรียบเทียบตัวอย่างที่ 5

ตัวอย่างที่ 5 แสดงให้เห็นถึงความสามารถในการแปลงซอสโค้ดที่มีรหัสดำเนินการแบบวนซ้ำตามเงื่อนไข และรหัสดำเนินการแบบแยกตามเงื่อนไข ซึ่งอยู่ภายในรหัสดำเนินการแบบวนซ้ำตามเงื่อนไข รวมทั้งเป็นตัวอย่างของการแปลงของซอสโค้ดที่มีรหัสดำเนินการที่มีเอ้าท์พุทในส่วนของ Resulting Indicator แสดงให้เห็นถึงความสามารถในการแปลงและตรวจสอบข้อความที่เกี่ยวข้องกับ Resulting Indicator และมีลักษณะของผังงานรวมทั้งคำอธิบายที่ถูกต้องตามซอสโค้ด รวมทั้ง

5.6.1 ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี

ข้อมูลนำเข้าซอสโค้ดอาร์พีจีตัวอย่างที่ 5 ดังภาพที่ 40 ประกอบด้วยรหัสดำเนินการ READ ซึ่งเป็นรหัสดำเนินการที่มีเอ้าท์พุทในส่วน Resulting Indicator โดยโปรแกรมกำหนดให้ Indicator 90 เป็นเอ้าพุทในการตรวจสอบความสามารถในการอ่านไฟล์ จะได้ว่า Indicator 90 มีสถานะเป็น

ON เมื่อไม่สามารถอ่านไฟล์ได้ และมีสถานะเป็น OFF เมื่อสามารถอ่านไฟล์ได้ จากซอสโค้ดจะใช้รหัสดำเนินการ DOWEQ ในการตรวจสอบสถานะของ Indicator 90

```
File  Edit  Format  View  Help
     FBH002P  IF  E           K        DISK
     FBH003P  UF  E           K        DISK
     F*----------------------------------------------------------------------
     C           KEY01     KLIST
     C                     KFLD           F0RTMD
     C                     KFLD           F0RLOT
     C                     KFLD           F0RUNT
     C           *LOVAL    SETLLBH002P
     C                     READ BH002P                   90
     C           *IN90     DOWEQ*OFF
     C           KEY01     CHAINBH003P              91
     C           *IN91     IFEQ *OFF
     C                     MOVELF0RID     ATCID
     C                     MOVELF0RSTS    ATCSTS
     C                     MOVELF0REPW    ATCEPW
     C                     MOVELF0REPD    ATCEPD
     C                     MOVELF0RENG    ATCENG
     C                     MOVEL'MANUAL'  ATCUPJ
     C                     UPDATBH003P
     C                     ENDIF
     C                     READ BH002P                   90
     C                     ENDDO
     C                     SETON                         LR
```

ภาพที่ 40 ซอสโค้ดอาร์พีจีตัวอย่างที่ 5

5.6.2 ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ

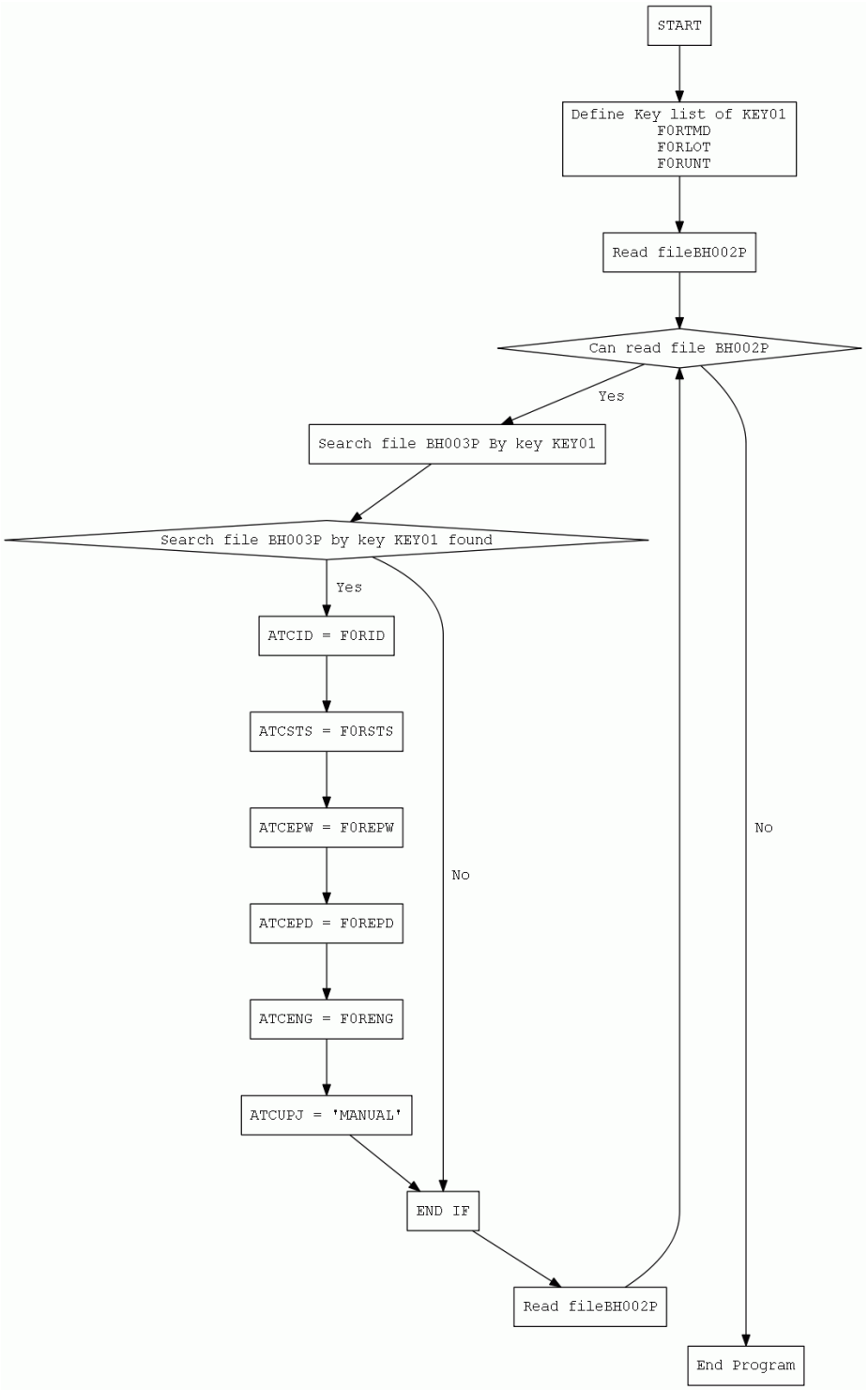
ผลลัพธ์ภาษากำกับเพิ่มดอทของซอสโค้ดตัวอย่างที่ 6 จะได้ดังภาพที่ 41

```
File  Edit  Format  View  Help
digraph G {
1 -> 2[label = "  " fontname = Courier ];
2 -> 3[label = "  " fontname = Courier ];
3 -> 4[label = "  " fontname = Courier ];
4 -> 5[label = " Yes " fontname = Courier ];
4 -> 15[label = " No " fontname = Courier ];
5 -> 6[label = "  " fontname = Courier ];
6 -> 7[label = " Yes " fontname = Courier ];
6 -> 13[label = " No " fontname = Courier ];
7 -> 8[label = "  " fontname = Courier ];
8 -> 9[label = "  " fontname = Courier ];
9 -> 10[label = "  " fontname = Courier ];
10 -> 11[label = "  " fontname = Courier ];
11 -> 12[label = "  " fontname = Courier ];
12 -> 13[label = "  " fontname = Courier ];
13 -> 14[label = "  " fontname = Courier ];
14 -> 4[label = "   " fontname = Courier ];
1[ label = "START" ,shape = Start fontname = Courier ];
2[ label = "Define Key list of KEY01\n F0RTMD\n F0RLOT\n F0RUNT" ,shape = rect fontname = Courier ];
3[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
4[ label = "Can read file BH002P" ,shape = diamond fontname = Courier ];
5[ label = "Search file BH003P By key KEY01" ,shape = rect fontname = Courier ];
6[ label = "Search file BH003P by key KEY01 found" ,shape = diamond fontname = Courier ];
7[ label = "ATCID = F0RID" ,shape = rect fontname = Courier ];
8[ label = "ATCSTS = F0RSTS" ,shape = rect fontname = Courier ];
9[ label = "ATCEPW = F0REPW" ,shape = rect fontname = Courier ];
10[ label = "ATCEPD = F0REPD" ,shape = rect fontname = Courier ];
11[ label = "ATCENG = F0RENG" ,shape = rect fontname = Courier ];
12[ label = "ATCUPJ = 'MANUAL'" ,shape = rect fontname = Courier ];
13[ label = "END IF" ,shape = rect fontname = Courier ];
14[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
15[ label = "End Program" ,shape = process fontname = Courier ];
{rank = sink ; 15}}
```

ภาพที่ 41 ผลลัพธ์ภาษากำกับเพิ่มดอทตัวอย่างที่ 5

5.6.3 ผลลัพธ์รูปภาพผังงาน

ผลที่ได้จากการนำเข้าภาษากำกับเพิ่มดอทตัวอย่างที่ 5 ไปสร้างแผนภาพผังงานด้วยโปรแกรมจินตทัศน์กราฟวิซเป็นรูปภาพ แสดงได้ดังภาพที่ 42



ภาพที่ 42 ผลลัพธ์รูปภาพผังงานตัวอย่างที่ 5

5.7 ผลการเปรียบเทียบตัวอย่างที่ 6

ตัวอย่างที่ 6 เป็นตัวอย่างของการแปลงของซอสโค้ดที่มีรหัสดำเนินการที่มีการทำงานแบบแยกตามเงื่อนไข ซึ่งมีการทำงานแบบแยกตามเงื่อนไขสองระดับ แสดงให้เห็นถึงความสามารถในการแปลงซอสโค้ดที่มีการทำงานแบบแยกตามเงื่อนไขในหลายระดับได้ถูกต้อง และมีลักษณะของผังงานรวมทั้งคำอธิบายที่ถูกต้องตามซอสโค้ด

5.7.1 ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี

ข้อมูลนำเข้าซอสโค้ดอาร์พีจีตัวอย่างที่ 6 ดังภาพที่ 43 ประกอบด้วยรหัสดำเนินการ IFEQ ซึ่งเป็นรหัสดำเนินการที่มีการทำงานแบบแยกตามเงื่อนไขในสองระดับ IFEQ ในระดับที่ 1 เป็นการตรวจสอบผลลัพธ์ของการค้นหาเรคคอร์ดภายในไฟล์ BH003P และ IFEQ ในระดับที่ 2 เป็นการตรวจสอบผลลัพธ์ของการค้นหาเรคคอร์ดภายในไฟล์ BH004P

File Edit Format View Help

```
     FBH002P  IF  E           K        DISK
     FBH003P  UF  E           K        DISK
     FBH004P  I   E           K        DISK
     F*----------------------------------------------------------------------
     C           KEY01     KLIST
     C                     KFLD           F0RTMD
     C                     KFLD           F0RLOT
     C                     KFLD           F0RUNT
     C           *LOVAL    SETLLBH002P
     C                     READ BH002P                   90
     C           *IN90     DOWEQ*OFF
     C           KEY01     CHAINBH003P              91
     C           *IN91     IFEQ *OFF
     C                     MOVELF0RID     ATCID
     C                     MOVELF0RSTS    ATCSTS
     C                     MOVELF0REPW    ATCEPW
     C                     MOVELF0REPD    ATCEPD
     C                     MOVELF0RENG    ATCENG
     C                     MOVEL'MANUAL'  ATCUPJ
     C           F0RTMD    CHAINBH004P              92
     C           *IN92     IFEQ *OFF
     C                     MOVELLOTSIZ    ATLOTS
     C                     ELSE
     C                     MOVEL*BLANK    ATLOTS
     C                     ENDIF
     C                     UPDATBH003P
     C                     ENDIF
     C                     READ BH002P                   90
     C                     ENDDO
     C                     SETON                         LR
```

ภาพที่ 43 ซอสโค้ดอาร์พีจีตัวอย่างที่ 6

5.7.2 ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ

ผลลัพธ์ภาษากำกับเพิ่มดอทของซอสโค้ดตัวอย่างที่ 6 จะได้ดังภาพที่ 44

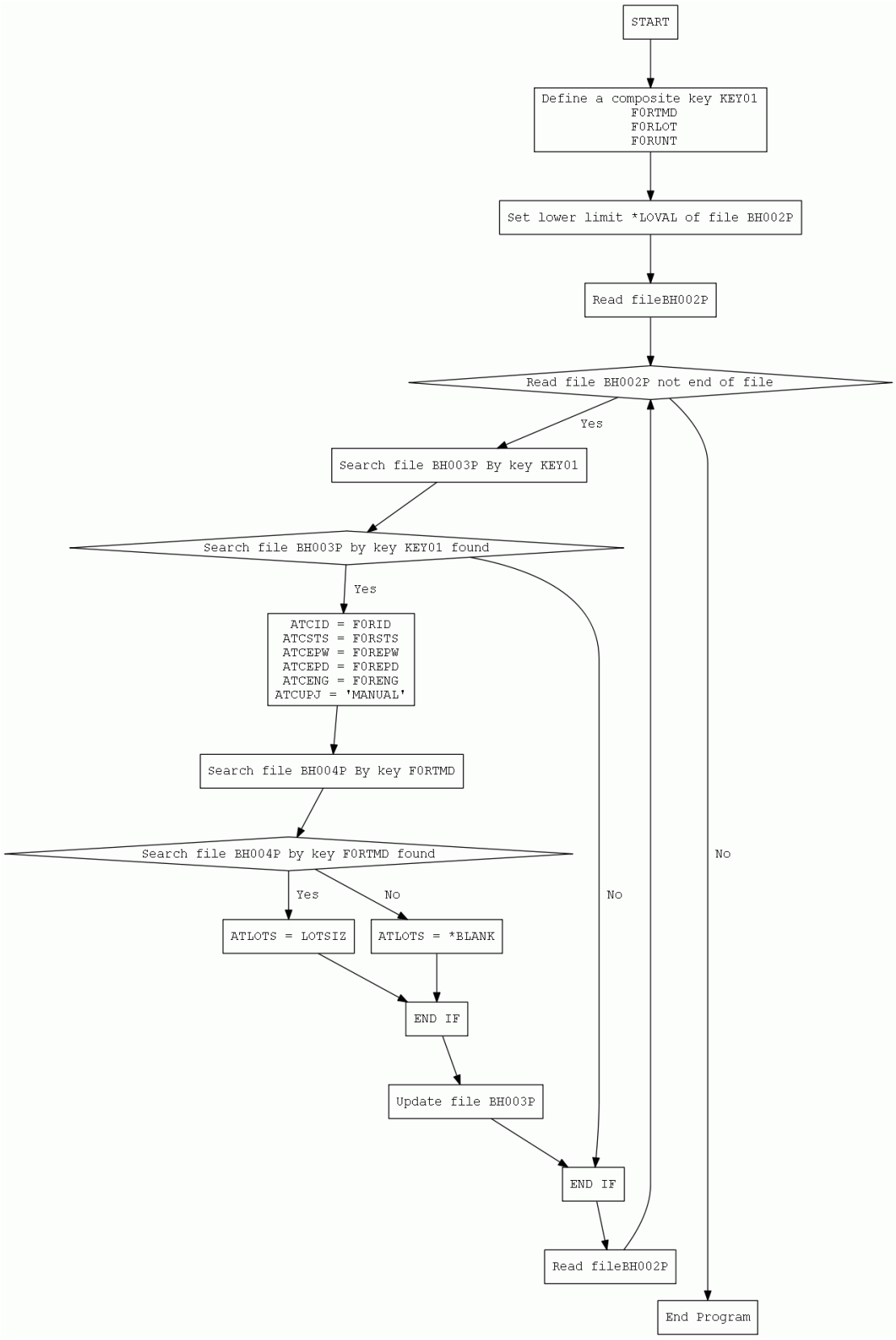
File Edit Format View Help

```
digraph G {
1 -> 2[label = "  " fontname = Courier ];
2 -> 3[label = "  " fontname = Courier ];
3 -> 4[label = "  " fontname = Courier ];
4 -> 5[label = "  " fontname = Courier ];
5 -> 6[label = " Yes " fontname = Courier ];
5 -> 18[label = " No " fontname = Courier ];
6 -> 7[label = "  " fontname = Courier ];
7 -> 8[label = " Yes " fontname = Courier ];
7 -> 16[label = " No " fontname = Courier ];
8 -> 9[label = "  " fontname = Courier ];
9 -> 10[label = "  " fontname = Courier ];
10 -> 11[label = " Yes " fontname = Courier ];
10 -> 13[label = " No " fontname = Courier ];
11 -> 14[label = "  " fontname = Courier ];
13 -> 14[label = "  " fontname = Courier ];
14 -> 15[label = "  " fontname = Courier ];
15 -> 16[label = "  " fontname = Courier ];
16 -> 17[label = "  " fontname = Courier ];
17 -> 5[label = "   " fontname = Courier ];
1[ label = "START" ,shape = Start fontname = Courier ];
2[ label = "Define a composite key KEY01\n F0RTMD\n F0RLOT\n F0RUNT" ,shape = rect fontname = Courier ];
3[ label = "Set lower limit *LOVAL of file BH002P" ,shape = rect fontname = Courier ];
4[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
5[ label = "Read file BH002P not end of file" ,shape = diamond fontname = Courier ];
6[ label = "Search file BH003P By key KEY01" ,shape = rect fontname = Courier ];
7[ label = "Search file BH003P by key KEY01 found" ,shape = diamond fontname = Courier ];
8[ label = "ATCID = F0RID\nATCSTS = F0RSTS\nATCEPW = F0REPW\nATCEPD = F0REPD\nATCENG = F0RENG\nATCUPJ = 'MANUAL'
9[ label = "Search file BH004P By key F0RTMD" ,shape = rect fontname = Courier ];
10[ label = "Search file BH004P by key F0RTMD found" ,shape = diamond fontname = Courier ];
11[ label = "ATLOTS = LOTSIZ" ,shape = rect fontname = Courier ];
13[ label = "ATLOTS = *BLANK" ,shape = rect fontname = Courier ];
14[ label = "END IF" ,shape = rect fontname = Courier ];
15[ label = "Update file BH003P" ,shape = rect fontname = Courier ];
16[ label = "END IF" ,shape = rect fontname = Courier ];
17[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
18[ label = "End Program" ,shape = process fontname = Courier ];
{rank = sink ; 18}}
```

ภาพที่ 44 ผลลัพธ์ภาษากำกับเพิ่มดอทตัวอย่างที่ 6

5.7.3 ผลลัพธ์รูปภาพผังงาน

ผลที่ได้จากการนำเข้าภาษากำกับเพิ่มดอทตัวอย่างที่ 6 ไปสร้างแผนภาพผังงานด้วยโปรแกรมจินตทัศน์กราฟวิซเป็นรูปภาพ แสดงได้ดังภาพที่ 45



ภาพที่ 45 ผลลัพธ์รูปภาพผังงานตัวอย่างที่ 6

5.8 ผลการเปรียบเทียบตัวอย่างที่ 7

ตัวอย่างที่ 7 เป็นตัวอย่างของการแปลงของซอสโค้ดที่มีรหัสดำเนินการที่มีการทำงานแบบวนซ้ำตามเงื่อนไข ซึ่งมีการทำงานแบบวนซ้ำตามเงื่อนไขสองระดับ แสดงให้เห็นถึงความสามารถในการแปลงซอสโค้ดที่มีการทำงานแบบวนซ้ำตามเงื่อนไขในหลายระดับได้ถูกต้อง และมีลักษณะของผังงาน รวมทั้งคำอธิบายที่ถูกต้องตามซอสโค้ด

5.8.1 ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี

ข้อมูลนำเข้าซอสโค้ดอาร์พีจีตัวอย่างที่ 7 ดังภาพที่ 46 ประกอบด้วยรหัสดำเนินการ DOWEQ ซึ่งเป็นรหัสดำเนินการที่มีการทำงานแบบซ้ำตามเงื่อนไขในสองระดับ DOWEQ ในระดับที่ 1 เป็นการตรวจสอบผลลัพธ์ของอ่านไฟล์ BH002P และ DOWEQ ในระดับที่ 2 เป็นการตรวจสอบผลลัพธ์ของการอ่านไฟล์ BH006P โดยจะมีการทำซ้ำเมื่อยังอ่านไฟล์ไม่จบไฟล์ แสดงให้เห็นถึงการทำงานแบบวนซ้ำตามเงื่อนไขในหลายระดับ รวมทั้งมีรหัสดำเนินการ IFEQ ภายใต้รหัสดำเนินการ DOWEQ ซึ่งแสดงให้เห็นถึงการทำงานแบบวนซ้ำตามเงื่อนไขภายใต้ การทำงานแบบแยกตามเงื่อนไข

File Edit Format View Help

```
     FBH002P  IF  E           K        DISK
     FBH003P  UF  E           K        DISK
     FBH006P  OF  E           K        DISK
     F*----------------------------------------------------------------------------
     C           KEY01     KLIST
     C                     KFLD           F0RTMD
     C                     KFLD           F0RLOT
     C                     KFLD           F0RUNT
     C           *LOVAL    SETLLBH002P
     C                     READ BH002P                   90
     C           *IN90     DOWEQ*OFF
     C           KEY01     CHAINBH003P              91
     C           *IN91     IFEQ *OFF
     C                     MOVELF0RID     ATCID
     C                     MOVELF0RSTS    ATCSTS
     C                     MOVELF0REPW    ATCEPW
     C                     MOVELF0REPD    ATCEPD
     C                     MOVELF0RENG    ATCENG
     C                     MOVEL'MANUAL'  ATCUPJ
     C                     Z-ADD0         NUMBER
     C           F0RID     READEBH006P                   92
     C           *IN92     DOWEQ*OFF
     C                     ADD  1         NUMBER
     C                     READEBH006P                   92
     C                     ENDDO
     C                     UPDATBH003P
     C                     ENDIF
     C                     READ BH002P                   90
     C                     ENDDO
     C                     SETON                         LR
```

ภาพที่ 46 ซอสโค้ดอาร์พีจีตัวอย่างที่ 7

5.8.2 ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ

ผลลัพธ์ภาษากำกับเพิ่มดอทของซอสโค้ดตัวอย่างที่ 7 จะได้ดังภาพที่ 47

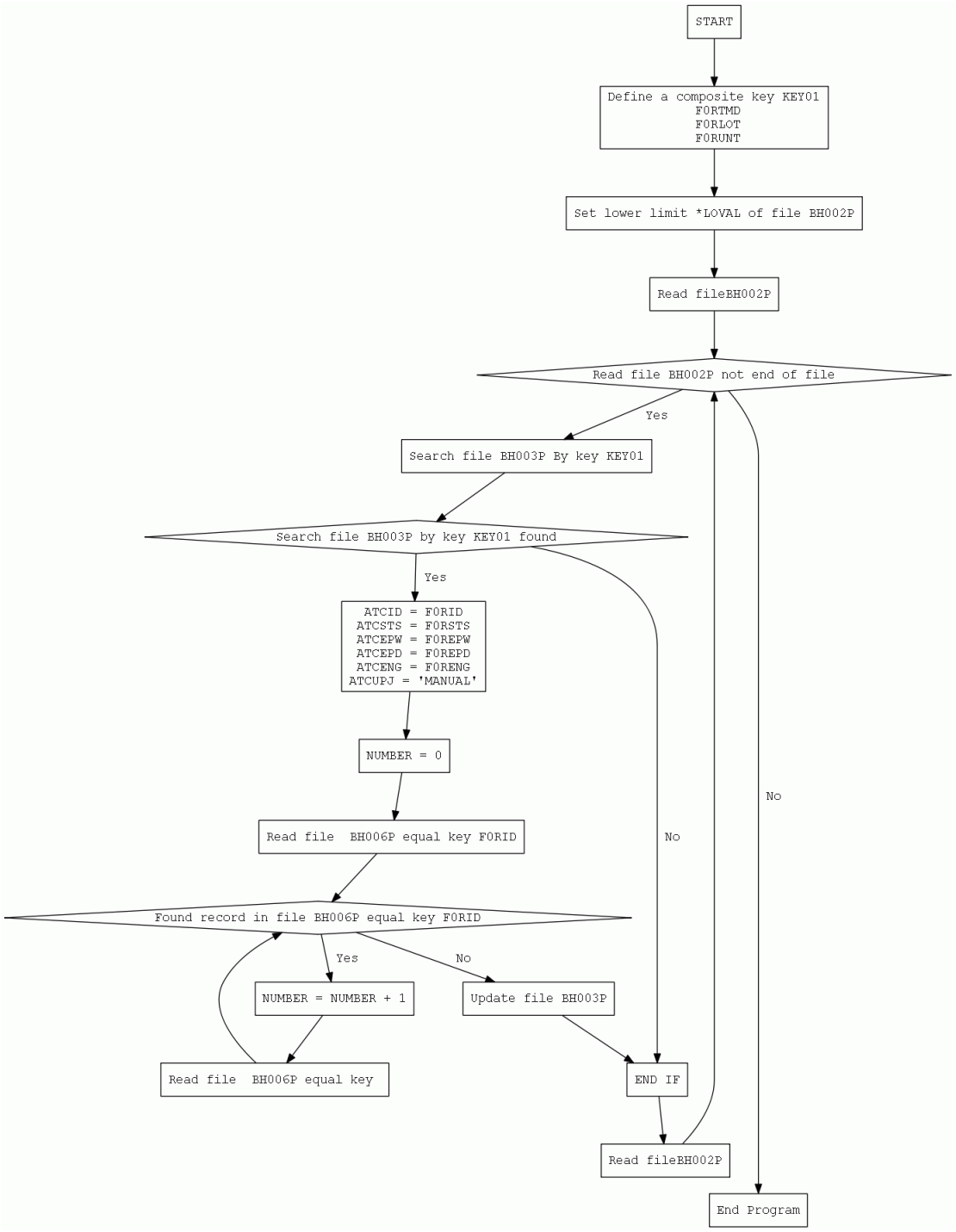
File Edit Format View Help

```
digraph G {
1 -> 2[label = "  " fontname = Courier ];
2 -> 3[label = "  " fontname = Courier ];
3 -> 4[label = "  " fontname = Courier ];
4 -> 5[label = "  " fontname = Courier ];
5 -> 6[label = " Yes " fontname = Courier ];
5 -> 17[label = " No " fontname = Courier ];
6 -> 7[label = "  " fontname = Courier ];
7 -> 8[label = " Yes " fontname = Courier ];
7 -> 15[label = " No " fontname = Courier ];
8 -> 9[label = "  " fontname = Courier ];
9 -> 10[label = "  " fontname = Courier ];
10 -> 11[label = "  " fontname = Courier ];
11 -> 12[label = " Yes " fontname = Courier ];
11 -> 14[label = " No " fontname = Courier ];
12 -> 13[label = "  " fontname = Courier ];
13 -> 11[label = "   " fontname = Courier ];
14 -> 15[label = "  " fontname = Courier ];
15 -> 16[label = "  " fontname = Courier ];
16 -> 5[label = "   " fontname = Courier ];
1[ label = "START" ,shape = Start fontname = Courier ];
2[ label = "Define a composite key KEY01\n F0RTMD\n F0RLOT\n F0RUNT" ,shape = rect fontname = Courier ];
3[ label = "Set lower limit *LOVAL of file BH002P" ,shape = rect fontname = Courier ];
4[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
5[ label = "Read file BH002P not end of file" ,shape = diamond fontname = Courier ];
6[ label = "Search file BH003P By key KEY01" ,shape = rect fontname = Courier ];
7[ label = "Search file BH003P by key KEY01 found" ,shape = diamond fontname = Courier ];
8[ label = "ATCID = F0RID\nATCSTS = F0RSTS\nATCEPW = F0REPW\nATCEPD = F0REPD\nATCENG = F0RENG\nATCUPJ = 'MANUAL'"
9[ label = "NUMBER = 0" ,shape = rect fontname = Courier ];
10[ label = "Read file  BH006P equal key F0RID" ,shape = rect fontname = Courier ];
11[ label = "Found record in file BH006P equal key F0RID" ,shape = diamond fontname = Courier ];
12[ label = "NUMBER = NUMBER + 1" ,shape = rect fontname = Courier ];
13[ label = "Read file  BH006P equal key " ,shape = rect fontname = Courier ];
14[ label = "Update file BH003P" ,shape = rect fontname = Courier ];
15[ label = "END IF" ,shape = rect fontname = Courier ];
16[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
17[ label = "End Program" ,shape = process fontname = Courier ];
{rank = sink ; 17}}
```

ภาพที่ 47 ผลลัพธ์ภาษากำกับเพิ่มดอทตัวอย่างที่ 7

5.8.3 ผลลัพธ์รูปภาพผังงาน

ผลที่ได้จากการนำเข้าภาษากำกับเพิ่มดอทตัวอย่างที่ 7 ไปสร้างแผนภาพผังงานด้วยโปรแกรมจินตทัศน์กราฟวิซเป็นรูปภาพ แสดงได้ดังภาพที่ 48



ภาพที่ 48 ผลลัพธ์รูปภาพผังงานตัวอย่างที่ 7

5.9 ผลการเปรียบเทียบตัวอย่างที่ 8

ตัวอย่างที่ 8 เป็นตัวอย่างของการแปลงของซอสโค้ดที่มีรหัสดำเนินการที่มีการทำงานแบบแยกตามเงื่อนไข ภายในการทำงานแบบวนซ้ำตามเงื่อนไข แสดงให้เห็นถึงความสามารถในการแปลงซอสโค้ดที่มีการทำงานแบบแยกตามเงื่อนไข ภายในการทำงานแบบวนซ้ำตามเงื่อนไขได้ถูกต้อง และมีลักษณะของผังงานรวมทั้งคำอธิบายที่ถูกต้องตามซอสโค้ด

5.9.1 ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี

ข้อมูลนำเข้าซอสโค้ดอาร์พีจีตัวอย่างที่ 8 ดังภาพที่ 49 ประกอบด้วยรหัสดำเนินการ DOWEQ และ IFEQ โดยมีรหัสดำเนินการ IFEQ ภายในรหัสดำเนินการ DOWEQ

File Edit Format View Help

```
     FBH002P  UF  E           K        DISK
     F*-------------------------------------------------------------------------
     C           *ENTRY    KLIST
     C                     KFLD           A0RTMD
     C           *LOVAL    SETLLBH002P
     C                     READ BH002P                   90
     C           *IN90     DOWEQ*OFF
     C           F0RTMD    IFEQ A0RTMD
     C                     MOVEL"30"      F0RSTS
     C                     UPDATBH002P
     C                     ENDIF
     C                     READ BH002P                   90
     C                     ENDDO
     C                     SETON                         LR
```

ภาพที่ 49 ซอสโค้ดอาร์พีจีตัวอย่างที่ 8

5.9.2 ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ

ผลลัพธ์ภาษากำกับเพิ่มดอทของซอสโค้ดตัวอย่างที่ 8 จะได้ดังภาพที่ 50

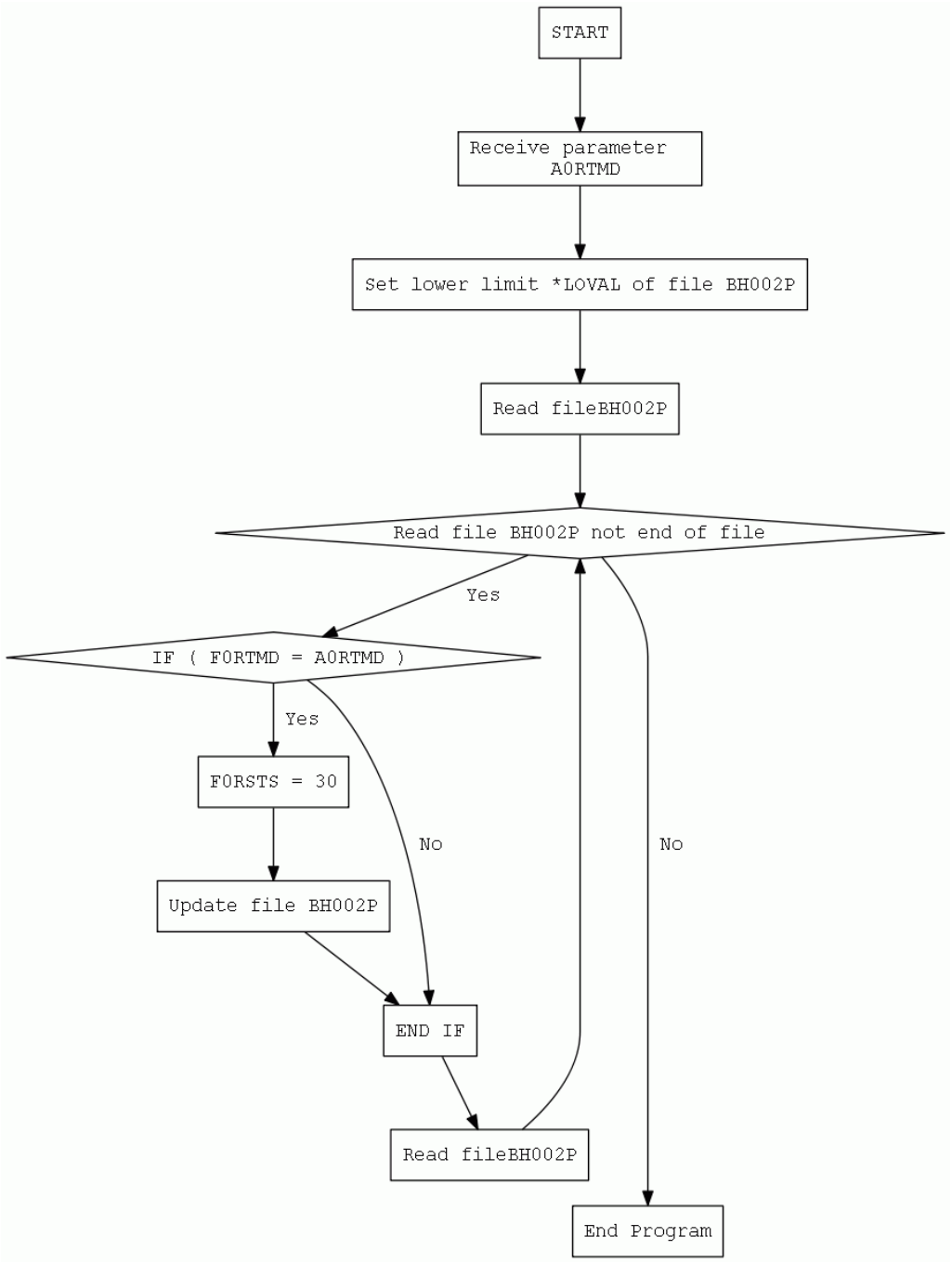
File Edit Format View Help

```
digraph G {
1 -> 2[label = "  " fontname = Courier ];
2 -> 3[label = "  " fontname = Courier ];
3 -> 4[label = "  " fontname = Courier ];
4 -> 5[label = "  " fontname = Courier ];
5 -> 6[label = " Yes " fontname = Courier ];
5 -> 11[label = " No " fontname = Courier ];
6 -> 7[label = " Yes " fontname = Courier ];
6 -> 9[label = " No " fontname = Courier ];
7 -> 8[label = "  " fontname = Courier ];
8 -> 9[label = "  " fontname = Courier ];
9 -> 10[label = "   " fontname = Courier ];
10 -> 5[label = "    " fontname = Courier ];
1[ label = "START" ,shape = Start fontname = Courier ];
2[ label = "Receive parameter  \n A0RTMD" ,shape = rect fontname = Courier ];
3[ label = "Set lower limit *LOVAL of file BH002P" ,shape = rect fontname = Courier ];
4[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
5[ label = "Read file BH002P not end of file" ,shape = diamond fontname = Courier ];
6[ label = " IF ( F0RTMD = A0RTMD )" ,shape = diamond fontname = Courier ];
7[ label = "F0RSTS = 30" ,shape = rect fontname = Courier ];
8[ label = "Update file BH002P" ,shape = rect fontname = Courier ];
9[ label = "END IF" ,shape = rect fontname = Courier ];
10[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
11[ label = "End Program" ,shape = process fontname = Courier ];
{rank = sink ; 11}}
```

ภาพที่ 50 ผลลัพธ์ภาษากำกับเพิ่มดอทตัวอย่างที่ 8

5.9.3 ผลลัพธ์รูปภาพผังงาน

ผลที่ได้จากการนำเข้าภาษากำกับเพิ่มดอทตัวอย่างที่ 8 ไปสร้างแผนภาพผังงานด้วยโปรแกรมจินตทัศน์กราฟวิซเป็นรูปภาพ แสดงได้ดังภาพที่ 51



ภาพที่ 51 ผลลัพธ์รูปภาพผังงานตัวอย่างที่ 8

5.10 ผลการเปรียบเทียบตัวอย่างที่ 9

ตัวอย่างที่ 9 เป็นตัวอย่างของการแปลงของซอสโค้ดที่มีรหัสดำเนินการที่มีการทำงานแบบวนซ้ำตามเงื่อนไข ภายในการทำงานแบบแยกตามเงื่อนไข แสดงให้เห็นถึงความสามารถในการแปลงซอสโค้ดที่มีการทำงานแบบวนซ้ำตามเงื่อนไข ภายในการทำงานแบบแยกตามเงื่อนไขได้ถูกต้อง และมีลักษณะของผังงานรวมทั้งคำอธิบายที่ถูกต้องตามซอสโค้ด

5.10.1 ข้อมูลนำเข้าซอสโค้ดภาษาอาร์พีจี

ข้อมูลนำเข้าซอสโค้ดอาร์พีจีตัวอย่างที่ 9 ดังภาพที่ 52 ประกอบด้วยรหัสดำเนินการ DOWEQ และ IFEQ โดยมีรหัสดำเนินการ DOWEQ ภายในรหัสดำเนินการ IFEQ

File Edit Format View Help

```
     FBH002P  UF  E           K        DISK
     FBM005P  IF  E           K        DISK
     F*----------------------------------------------------------------------
     C           *ENTRY    PLIST
     C                     PARM           A0RTMD
     C           A0RTMD    CHAINBM005P              91
     C           *IN91     IFEQ *OFF
     C           A0RTMD    SETLLBH002P
     C                     READ BH002P                    90
     C           F0RTMD    DOWEQA0RTMD
     C                     MOVEL1         F0RFLG
     C                     UPDATBH002P
     C                     READ BH002P                    90
     C                     ENDDO
     C                     ENDIF
     C                     SETON                          LR
```

ภาพที่ 52 ซอสโค้ดอาร์พีจีตัวอย่างที่ 9

5.10.2 ผลลัพธ์ภาษากำกับเพิ่มดอทของระบบ

ผลลัพธ์ภาษากำกับเพิ่มดอทของซอสโค้ดตัวอย่างที่ 9 จะได้ดังภาพที่ 53

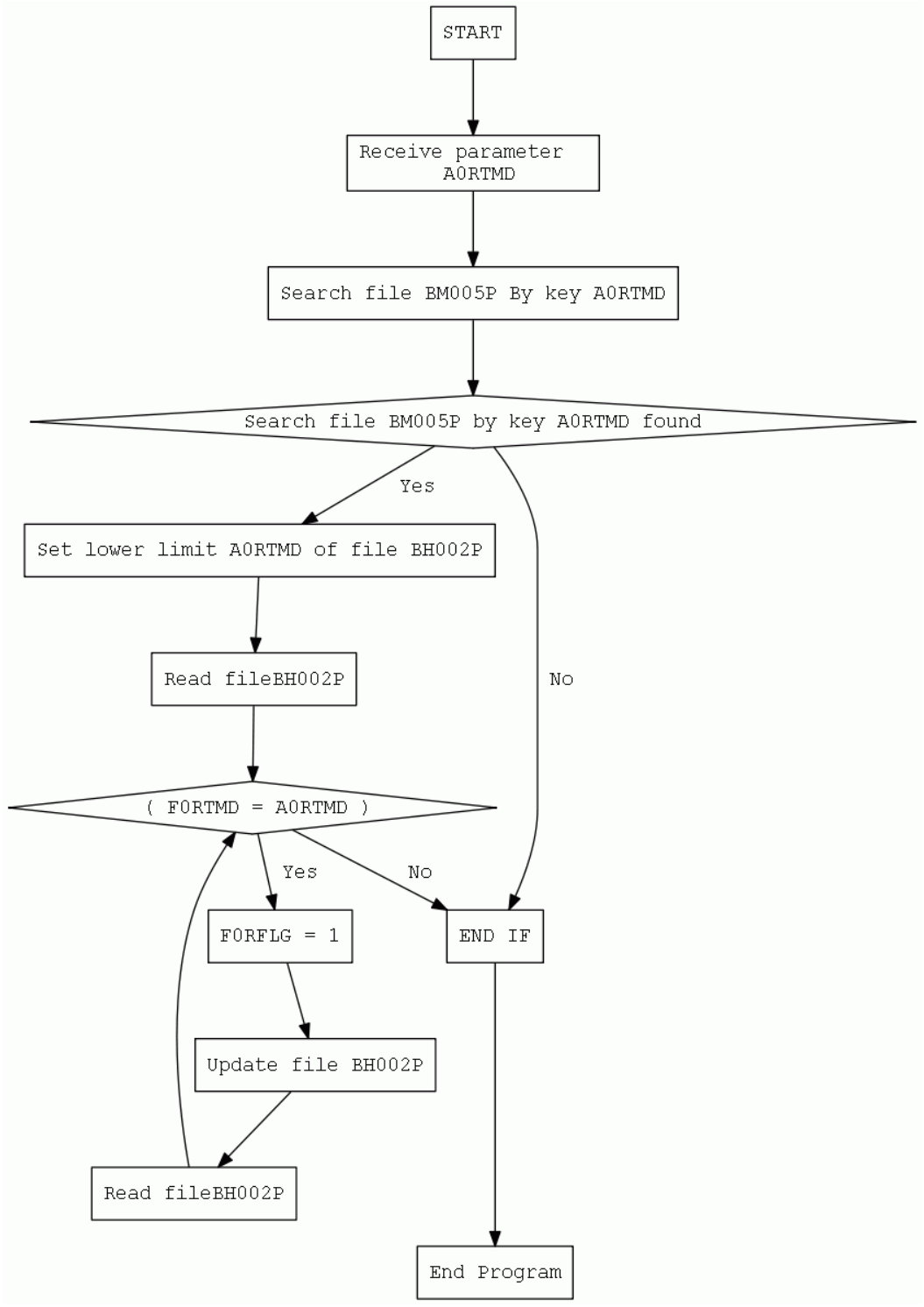
File Edit Format View Help

```
digraph G {
1 -> 2[label = "  " fontname = Courier ];
2 -> 3[label = "  " fontname = Courier ];
3 -> 4[label = "  " fontname = Courier ];
4 -> 5[label = " Yes " fontname = Courier ];
4 -> 11[label = " No " fontname = Courier ];
5 -> 6[label = "  " fontname = Courier ];
6 -> 7[label = "  " fontname = Courier ];
7 -> 8[label = " Yes " fontname = Courier ];
7 -> 11[label = " NO " fontname = Courier ];
8 -> 9[label = "  " fontname = Courier ];
9 -> 10[label = "  " fontname = Courier ];
10 -> 7[label = "   " fontname = Courier ];
11 -> 12[label = "  " fontname = Courier ];
1[ label = "START" ,shape = Start fontname = Courier ];
2[ label = "Receive parameter  \n A0RTMD" ,shape = rect fontname = Courier ];
3[ label = "Search file BM005P By key A0RTMD" ,shape = rect fontname = Courier ];
4[ label = "Search file BM005P by key A0RTMD found" ,shape = diamond fontname = Courier ];
5[ label = "Set lower limit A0RTMD of file BH002P" ,shape = rect fontname = Courier ];
6[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
7[ label = " ( F0RTMD = A0RTMD )" ,shape = diamond fontname = Courier ];
8[ label = "F0RFLG = 1" ,shape = rect fontname = Courier ];
9[ label = "Update file BH002P" ,shape = rect fontname = Courier ];
10[ label = "Read fileBH002P" ,shape = rect fontname = Courier ];
11[ label = "END IF" ,shape = rect fontname = Courier ];
12[ label = "End Program" ,shape = process fontname = Courier ];
{rank = sink ; 12}}
```

ภาพที่ 53 ผลลัพธ์ภาษากำกับเพิ่มดอทตัวอย่างที่ 9

5.10.3 ผลลัพธ์รูปภาพผังงาน

ผลที่ได้จากการนำเข้าภาษากำกับเพิ่มดอทตัวอย่างที่ 9 ไปสร้างแผนภาพผังงานด้วยโปรแกรมจินตทัศน์กราฟวิซเป็นรูปภาพ แสดงได้ดังภาพที่ 54



ภาพที่ 54 ผลลัพธ์รูปภาพผังงานตัวอย่างที่ 9

บทที่ 6
สรุปผลการวิจัย และข้อเสนอแนะ

6.1 สรุปผลการวิจัย

งานวิจัยนี้ได้นำเสนอวิธีการ และเครื่องมือดึงความรู้จากซอสโค้ดภาษาอาร์พีจีมาสร้างเป็นผังงานแสดงรายละเอียดขั้นตอนการทำงาน เพื่อลดระยะเวลา ความพยายามที่ใช้ในการทำความเข้าใจซอสโค้ด และการจัดทำเอกสารการออกแบบ รวมทั้งลดข้อบกพร่องในการแก้ไขหรือปรับปรุงระบบเก่าแก่ โดยที่เครื่องมือจะได้ผลลัพธ์เป็นภาษากำกับเพิ่มดอท ที่สามารถนำเข้าสู่โปรแกรมจินตทัศน์กราฟวิช เพื่อให้ได้ผลลัพธ์เป็นรูปภาพของแผนภาพผังงาน

ซึ่งจากการทดลองใช้งานเครื่องมือสร้างผังงานในบทที่ 5 แสดงให้เห็นว่าเครื่องมือสามารถแปลงซอสโค้ดภาษาอาร์พีจีเป็นภาษากำกับเพิ่มดอทได้ถูกต้อง และสามารถนำเข้าภาษากำกับเพิ่มดอทเพื่อสร้างรูปภาพผังงานแสดงการทำงานของซอสโค้ดอาร์พีจี จากการเรียกใช้งานโปรแกรมจินตทัศกราฟวิซได้ถูกต้อง

6.2 ข้อจำกัด

1. งานวิจัยนี้ยังไม่สามารถรองรับการสร้างแผนภาพผังงานจากซอสโค้ดที่มีมากกว่าหนึ่งไฟล์ได้
2. งานวิจัยนี้รองรับการแปลงแผนภาพผังงานของซอสโค้ดอาร์พีจี/400

6.3 แนวทางการวิจัยต่อ

- การแปลงซอสโค้ดเป็นภาษากำกับเพิ่ม ยังรองรับเพียงแค่ภาษากำกับเพิ่มดอทเท่านั้น ซึ่งในปัจจุบันมีภาษากำกับเพิ่มที่สามารถสร้างแผนภาพผังงานงานอีกหลายประเภท เช่น GXL (Graph eXchange Language) และ GraphML เป็นต้น
- การแปลงภาษากำกับเพิ่มดอทเป็นรูปภาพนั้นใช้การนำเข้าภาษากำกับเพิ่มดอทสู่โปรแกรมจินตทัศน์กราฟวิช ซึ่งรูปภาพที่ได้จะเกิดจากการจัดรูปแบบด้วยโปรแกรมจินตทัศน์ ทำให้ไม่สามารถจัดตำแหน่ง และรูปแบบของผังงานได้ตามต้องการ

การแปลงซอสโค้ดเป็นแผนภาพผังงาน ยังรองรับเพียงแค่การแปลงซอสโค้ดอาร์พีจี/400 เท่านั้น ซึ่งในปัจจุบันมีซอสโค้ดภาษาอาร์พีจีหลายเวอร์ชัน

รายการอ้างอิง

1. IBM, RPG/400 User's Guide. 3 ed. 1994, Canada: IBM Canada Ltd. Laboratory.
2. Vasudevan, B.G., S. Dhanapanichkul, and R. Balakrishnan. Flowchart knowledge extraction on image processing. in IJCNN. 2008. IEEE.
3. Winter, A. Exchanging Graphs with GXL. in Graph Drawing - 9th International Symposium. 2001. Springer Verlag.
4. Research, A.T. and amp, Graphviz - Graph Visualization Software. 2008.
5. Gansner, E., E. Koutsofios, and S. North, Drawing graphs with dot. 2006.
6. Contributors, B.N.a. Welcome to JGraphT. 2011 [cited 2014 18]; Available from: http://jgrapht.org/.

ภาคผนวก

ภาคผนวก ก

ตัวอย่างรหัสดำเนินการ และข้อความกำกับโหนดของกราฟแบบมีทิศทาง

ตารางที่ 5 ข้อความกำกับโหนดของกราฟแบบมีทิศทาง

รหัสดำเนินการ	**ข้อความ**
ADD (Add)	Case Factor 1 = Blank [Result Field] = [Result Field] + [Factor 2] Case Factor 1 <> Blank [Result Field] = [Factor 1] + [Factor 2]
ANDxx (And)	ANDEQ And ([Factor 1] = [Factor 2]) ANDGT And ([Factor 1] > [Factor 2]) ANDLT And ([Factor 1] < [Factor 2]) ANDGE And ([Factor 1] >= [Factor 2]) ANDLE And ([Factor 1] <= [Factor 2])
BEGSR (Beginning of Subroutine)	-

CABxx (Compare and Branch)	CABEQ
	IF ([Factor 1]= [Factor 2])
	Add new node
	Go to tag [Result Field]
	
	CABGT
	IF ([Factor 1] > [Factor 2])
	Add new node
	Go to tag [Result Field]
	
	CABLT
	IF ([Factor 1] < [Factor 2])
	Add new node
	Go to tag [Result Field]
	
	CABGE
	IF ([Factor 1] >= [Factor 2])
	Add new node
	Go to tag [Result Field]
	
	CABLE
	IF ([Factor 1] <= [Factor 2])
	Add new node
	Go to tag [Result Field]

CALL (Call a Program)	Call program [Factor 2]

CASxx (Conditionally Invoke Subroutine)	<u>CASEQ</u>
	IF ([Factor 1]= [Factor 2])
	Add new node
	Execute [Result Field]
	
	<u>CASGT</u>
	IF ([Factor 1] > [Factor 2])
	Add new node
	Execute [Result Field]
	
	<u>CASLT</u>
	IF ([Factor 1] < [Factor 2])
	Add new node
	Execute [Result Field]
	
	<u>CASGE</u>
	IF ([Factor 1] >= [Factor 2])
	Add new node
	Execute [Result Field]
	
	<u>CASLE</u>
	IF ([Factor 1] <= [Factor 2])

CHAIN (Random Retrieval from a File)	Search File [Factor 2] by key [Factor 1]
CLEAR (Clear)	Clear [Factor 2]
CLOSE (Close Files)	Close file [Factor 2]
COMP (Compare)	Compare [Factor 1] and [Factor 2]
DEFN (Field Definition)	Define field [Factor 3] same as field [Factor 2]

DELET (Delete Record)	Delete record of file [Factor 2]
DIV (Divide)	[Result Field] = [Factor1] / [Factor 2]

DOUxx (Do Until)	DOUEQ
	IF ([Factor 1]= [Factor 2])
	
	DOUGT
	IF ([Factor 1] > [Factor 2])
	
	DOULT
	IF ([Factor 1] < [Factor 2])
	
	DOUGE
	IF ([Factor 1] >= [Factor 2])
	
	DOULE
	IF ([Factor 1] <= [Factor 2])

DOWxx (Do While)	DOWEQ
	IF ([Factor 1]= [Factor 2])
	
	DOWGT
	IF ([Factor 1] > [Factor 2])
	
	DOWLT
	IF ([Factor 1] < [Factor 2])
	
	DOWGE
	IF ([Factor 1] >= [Factor 2])
	
	DOWLE
	IF ([Factor 1] <= [Factor 2])

ELSE (Else)	Else
ENDyy (End a Group)	End
ENDSR (End of Subroutine)	-
EXFMT (Write/Then Read Format)	Write and Read recorder format [Factor 2]
EXSR (Invoke Subroutine)	Execute subroutine [Factor 3]
FEOD (Force End of Data)	Force End of data [Factor 2]
FORCE (Force a Certain File to Be Read Next Cycle)	Force file [Factor 2] to be read next cycle
GOTO (Go To)	-

IFxx (If)	IFEQ
	IF ([Factor 1] = [Factor 2])
	
	IFGT
	IF ([Factor 1] > [Factor 2])
	
	IFLT
	IF ([Factor 1] < [Factor 2])
	
	IFGE
	IF ([Factor 1] >= [Factor 2])
	
	IFLE
	IF ([Factor 1] <= [Factor 2])

IN (Retrieve a Data Area)	Retrieve data area [Factor 2]

KFLD (Define Parts of a Key)	[Result Field]
KLIST (Define a Composite Key)	Define a composite key [Factor 1]
LEAVE (Leave a Do Group)	Leave
LOKUP (Look Up)	Search [Factor 1] in [Factor 2]
MOVE (Move)	[Result Field] = [Factor 2]
MOVEA (Move Array)	Array [Result Field] = [Factor 2]
MOVEL (Move Left)	[Result Field] = [Factor 2]
MULT (Multiply)	[Result Field] = [Factor1] X [Factor 2]
OPEN (Open File for Processing)	Open File [Factor 2]
ORxx (Or)	OREQ Or ([Factor 1] = [Factor 2]) ORGT Or ([Factor 1] > [Factor 2]) ORLT Or ([Factor 1] < [Factor 2]) ORGE Or ([Factor 1] >= [Factor 2]) ORLE Or ([Factor 1] <= [Factor 2])

OTHER (Otherwise Select)	Other
PARM (Identify Parameters)	[Result Field]
PLIST (Identify a Parameter List)	Case factor 1 = '*ENTRY' Receive parameter Case other Identify a parameter list [Factor 1]
READ (Read a Record)	Read file [Factor 2]
READC (Read Next Changed Record)	Read next changed record of file [Factor 2]
READE (Read Equal Key)	Read file [Factor 2] equal key [Factor 1]
READP (Read Prior Record)	Read prior record of file [Factor 2]
REDPE (Read Prior Equal)	Read file [Factor 2] prior equal key [Factor 1]
RETRN (Return to Caller)	Return to the caller
SELEC (Begin a Select Group)	Select
SETGT (Set Greater Than)	Set greater than [Factor] of file [Factor 2]
SETLL (Set Lower Limit)	Set lower limit [Factor 1] of file [Factor 2]
SETOF (Set Off)	Set off indicator [Indicator1], [Indicator 2], [Indicator 3]
SETON (Set On)	Set on indicator [Indicator1], [Indicator 2], [Indicator 3]
SORTA (Sort an Array)	Sort an Array [Factor 2]

SQRT (Square Root)	[Result Field] = [Factor1] / [Factor 2]
SUB (Subtract)	[Result Field] = [Factor1] - [Factor 2]
TAG (Tag)	-
UNLCK (Unlock a Data Area or Release a Record)	Unlock [Factor 2]
UPDAT (Modify Existing Record)	Update file [Factor 2]

WHxx (When True Then Select)	WHEQ
	IF ([Factor 1] = [Factor 2])
	
	WHGT
	IF ([Factor 1] > [Factor 2])
	
	WHLT
	IF ([Factor 1] < [Factor 2])
	
	WHGE
	IF ([Factor 1] >= [Factor 2])
	
	WHLE
	IF ([Factor 1] <= [Factor 2])

WRITE (Create New Records)	Write record in file [Factor 2]
XFOOT (Summing the Elements of an Array)	[Result Field] = Sum of array [Factor 2]
Z-ADD (Zero and Add)	[Result Field] = [Factor 2]
Z-SUB (Zero and Subtract)	[Result Field] = - [Factor 2]

ภาคผนวก ข

ตัวอย่างรหัสดำเนินการ และข้อความของ Resulting Indicator

ตารางที่ 6 ข้อความของ Resulting Indicator

รหัสดำเนินการ	**ตำแหน่งของ Indicator**	**ข้อความ**	
		เมื่อ Indicator มีสถานะเป็น ON	**เมื่อ Indicator มีสถานะเป็น OFF**
CALL (Call a Program)	54-55	-	-
	56-57	Call program [Program Name] has error return	Call program [Program Name] complete
	58-59	-	-
CHAIN (Random Retrieval from a File)	54-55	Search Key [Key Name] in file [File Name] Not found	Search Key [Key Name] in file [File Name] found
	56-57	Search operation Key [Key Name] in file [File Name] not complete	Search operation Key [Key Name] in file [File Name] complete
	58-59	-	-
CLOSE (Close Files)	54-55	-	-
	56-57	Close file [File Name] completed	Close file [File Name] not completed
	58-59	-	-
DELET (Delete Record)	54-55	Record to be deleted in file [File name] by key [Key Name] is not found	Record to be deleted in file [File name] by key [Key Name] is found
	56-57	Delete operation by Key [Key Name] in file [Fine Name] not complete	Delete operation by Key [Key Name] in file [Fine Name] complete
	58-59	-	-
EXFMT	54-55	-	-

(Write/Then Read Format)	56-57	Write/Then Read Format [Record Format Name] is not completed	Write/Then Read Format [Record Format Name] is completed
	58-59	-	-
FEOD (Force End of Data)	54-55	-	-
	56-57	Force End of Data [File Name] is not completed	Force End of Data [File Name] is completed
	58-59	-	-
IN (Retrieve a Data Area)	54-55	-	-
	56-57	An error occurs during retrieve data area [Data area name]	Retrieve a Data Area [Data area name] is completed
	58-59	-	-
OPEN (Open File for Processing)	54-55	-	-
	56-57	Open file [File name] is not successful	Open file [File name] is successful
	58-59	-	-
OUT (Write a Data Area)	54-55	-	-
	56-57	An error occurs during write data area [Data area name]	Write data area [Data area name] completed
	58-59	-	-
READ (Read a Record)	54-55	-	-
	56-57	Read file [File Name] complete	Read file [File Name] not complete
	58-59	Read file [File Name] end of file	Read file [File Name] not end of file
UPDAT	54-55	-	-

(Modify Existing Record)	56-57	Update file [File Name] not complete	Update file [File Name] complete
	58-59	-	-
WRITE (Create New Records)	54-55	-	-
	56-57	Write file [File Name] complete	Write file [File Name] not complete
	58-59	-	-

ประวัติผู้เขียนวิทยานิพนธ์

นางสาวกชพร สันติปารคู เกิดเมื่อวันที่ 19 ธันวาคม พ.ศ. 2531 ที่จังหวัดขอนแก่น สำเร็จการศึกษาปริญญาตรีหลักสูตรวิศวกรรมศาสตรบัณฑิต (วศ.บ.) สาขาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ สถาบันมหิดล ในปีการศึกษา 2554 และเข้าศึกษาต่อในหลักสูตรวิทยาศาสตรมหาบัณฑิต สาขาวิศวกรรมซอฟต์แวร์ ที่ภาควิชาวิศวกรรมคอมพิวเตอร์ คณะวิศวกรรมศาสตร์ จุฬาลงกรณ์มหาวิทยาลัย ในปีการศึกษา 2555