เคล็ดลับในการเรียนภาษาคอมพิวเตอร์

โดย รศ. ดร. ครรชิต มาลัยวงศ์

จากนิตยสาร Business Computer Magazine
ฉบับเดือน มีนาคม 1994

นักเทคโนโลยีด้านคอมพิวเตอร์เชื่อกันว่าในอนาคตการใช้คอมพิวเตอร์จะง่ายมากกว่าในปัจจุบัน โดยเฉพาะการสั่งงานคอมพิวเตอร์อาจใช้ภาษาธรรมชาติ เช่น ภาษาอังกฤษหรือภาษาไทยแทนการใช้ภาษาปาสกาล เมื่อถึงยุคนั้นพวกเราคงจะสบาย อยากสั่งคอมพิวเตอร์ให้คิดงานอะไรก็พูดบอกเป็นภาษาที่เราใช้กันอยู่นั่นแหละครับ จะพูดเพราะแบบมีจ๊ะจ๋าหรือจะมึงวาพาโวยอย่างไรก็ได้

ขณะนี้นักวิจัยคอมพิวเตอร์ไทยในห้องปฏิบัติการหลายแห่งกำลังศึกษาวิจัยโครงสร้างของภาษาไทยเป็นการใหญ่ ยกตัวอย่างง่ายๆ เช่น พจนานุกรมอิเล็กทรอนิกส์ที่มีขายอยู่เวลานี้ ก็เป็นส่วนหนึ่งของการวิจัยที่ว่านี้ จริงอยู่งานวิจัยเหล่านี้อาจจะยังไม่มีวัตถุประสงค์ที่จะทำให้คอมพิวเตอร์รู้ภาษาไทย แต่ผลงานก็จะเป็นประโยชน์ต่อไปในอนาคต

มีคำถามอยู่ว่า ถ้าหากในอนาคตคอมพิวเตอร์จะรู้ภาษาคนแล้วละก็ เราจะต้องเรียนรู้ภาษาคอมพิวเตอร์ไปทำไม เรื่องอะไรจะต้องบรรจุภาษา Pascal ภาษา C ภาษา Cobol ไว้ในหลักสูตรตั้งแต่ระดับมัธยมขึ้นมาด้วยเล่า คำตอบก็คือ ยังอีกนานครับกว่าจะถึงยุคคอมพิวเตอร์รู้ภาษาคน หากเราจะรอไปถึงยุคนั้นโดยไม่ยอมเรียนรู้อะไร หรือทำอะไรเลยก็จะเป็นการเสียโอกาสไปเปล่าๆ

นับตั้งแต่ผมเริ่มหัดเขียนโปรแกรมเมื่อสามสิบปีก่อนมาจนกระทั่งบัดนี้ ผมมีความรู้สึกว่าการเขียนโปรแกรมเป็นงานที่ท้าทายสติปัญญามากที่สุดอย่างหนึ่ง แม้ผมจะเลิกราการเขียนโปรแกรมไปแล้ว แต่ก็ยังมีโอกาสแนะนำวิธีการเขียน และการศึกษาด้านโปรแกรมและซอฟต์แวร์อยู่โดยตลอด เท่าที่ผ่านมา ผมพบว่าภาษาคอมพิวเตอร์โดยเฉพาะภาษาโปรแกรมมีเสน่ห์น่าเรียนรู้และยังมีเรื่องท้าทายอีกมาก ยิ่งมีหลักการและแนวคิดใหม่ ๆ อย่างเช่นการเขียนโปรแกรมตรรกะ (Logic Programming) และหลักการเชิงวัตถุ (Object Oriented Concept) ออกมาใหม่ ๆ ด้วยแล้วยิ่งน่าสนใจ และน่าเรียนรู้มากกว่าสมัยที่ผมเรียน Fortran II , Autocode หรือ Algol เป็นทวีคูณเลยทีเดียว

ใจจริงผมยังคิดว่าภาษาคอมพิวเตอร์นั้นยังไม่มีวันตายหรอกครับ เพราะจะอย่างไร ๆ ก็ไม่น่าจะมีใครมีความสามารถคิดวงจรอิเล็กทรอนิกส์ที่วิเคราะห์คำสั่งภาษาไทยได้ คงจะต้องเป็นคำสั่งภาษาเครื่องเหมือนคอมพิวเตอร์สมัยนี้อยู่ดี ถ้าเช่นนั้นเราก็จะต้องมีภาษาที่อยู่ระหว่างภาษาเครื่องกับภาษาคนอยู่ดี และภาษานั้นคงไม่แคล้วเป็นภาษาโปรแกรมอย่างเช่น ภาษา C หรือ C++ นั่นแหละครับ

เวลานี้ใคร ๆ ก็อยากเรียนรู้คอมพิวเตอร์ แต่เรียนเพื่อจะใช้นะครับ ไม่มีใครอยากเรียนเพื่อจะสร้างโปรแกรมเลย คนที่เรียนคอมพิวเตอร์ตั้งแต่นักเรียนมัธยมไปจนถึงนักศึกษาปริญญาโทนั้นพอบอกว่าจะต้องเรียนการเขียนโปรแกรม หรือต้องเขียนโปรแกรมเท่านั้นเป็นหัวหดไปหมด เพราะกลัวจริง ๆ เรื่องเขียนโปรแกรมนี่ ผมพยายามคิดว่าเกิดอะไรขึ้นหนอถึงได้กลัวการเขียนโปรแกรมนัก คำตอบที่ผมพอจะนึกได้ก็คือ

- การเขียนโปรแกรมเหมือนกับการใช้ภาษาอื่น คนไทยเป็นชาติที่ไม่ค่อยเก่งภาษาอื่นเท่าใดนัก เช่น อุตส่าห์เล่าเรียนภาษาอังกฤษเป็นวรรคเป็นเวรตั้งแปดปีสิบปี แต่พูดไม่ได้หรอกครับ เจอฝรั่งมานี่เราเดินหนีเลย แทนที่จะวิ่งเข้าไปสนทนาด้วย
- เรียนการเขียนโปรแกรมแล้วไม่น่าเข้าใจ เพราะภาษาเขียนโปรแกรมเป็นรูปแบบที่เราไม่ชินการสั่งงานเป็นขั้นตอนก็ทำไม่ได้ เพราะเรารู้จักแต่การเดาคำตอบต่าง ๆ ยิ่งกว่าจะวิเคราะห์กระบวนการหรือเงื่อนไขของปัญหา ในเมื่อการเขียนโปรแกรมด้วยภาษาโปรแกรมจะต้องรู้จักไล่เลียงและเรียบเรียงความคิดให้เป็นขั้นตอนอย่างถูกต้อง คนไทยจึงทำไม่ได้
- อาจารย์ผู้สอนวิธีการเขียนโปรแกรมไม่มีความสามารถพอ ตรงนี้ผมคิดว่าเป็นเรื่องสำคัญมาก เวลานี้ยังไม่เคยมีการสัมมนาแลกเปลี่ยนความคิดเห็นเกี่ยวกับการสอนวิธีเขียนโปรแกรมกันเลย อาจารย์หลายคนอาจเคยเขียนเป็นแต่โปรแกรมง่าย ๆ ไม่เคยเขียนโปรแกรมใหญ่ ๆ กันเลย จริงอยู่อาจารย์หลายคนอาจแม่นในด้านคำสั่ง รูปแบบ และไวยกรณ์ของคำสั่งในภาษาโปรแกรม แต่นั่นไม่ได้รับประกันว่าท่านจะเขียนโปรแกรมได้ หรือสอนวิธีเขียนโปรแกรมได้
- ผู้เรียนไม่รู้วิธีเรียนที่ถูกต้อง การเรียนเรื่องเขียนโปรแกรมนี้จำเป็นต้องมีเทคนิคที่ดี และจะต้องฝึกฝนเขียนโปรแกรมอย่างมีระบบ ลำพังมานั่งฟังอาจารย์สอนและเขียนโปรแกรมเพียงสองสามโปรแกรมต่อเทอมอย่างที่ทำกันอยู่เวลานี้ ไม่มีทางเก่งด้านโปรแกรมได้หรอกครับ

การเรียนภาษาคอมพิวเตอร์ใด ๆ ให้เก่งนั้น ไม่ใช่เรื่องยากเย็นอะไรหรอกครับ ที่กลัว ๆ กันก็เพราะไม่เคยมีใครมาสอนให้แก้ปัญหาที่กล่าวไปแล้ว ดังนั้นเวลาเรียนแล้วไม่เข้าใจ เขียนโปรแกรมแล้วไม่ทำงานอย่างที่ต้องการ จึงเกิดความท้อถอยเข็ดขยาดเอาดื้อ ๆ

ผมขอแยกเนื้อหาเกี่ยวกับภาษาโปรแกรมออกเป็นส่วน ๆ ดังนี้

- คำสั่งและรูปแบบของคำสั่ง
- แนวคิดเกี่ยวกับหลักการทำงานของคำสั่งและโปรแกรม
- โครงสร้างของโปรแกรม
- แนวคิดเกี่ยวกับขั้นตอนการแก้ปัญหา
- การตรวจสอบแก้ไขความผิดพลาดในโปรแกรม

ในบรรดาเนื้อหาเหล่านี้มีเรื่องเดียวที่เป็นเรื่องกว้าง ๆ ไม่ขึ้นอยู่กับภาษาโปรแกรมใด ๆ นั่นก็คือแนวคิดเกี่ยวกับขั้นตอนการแก้ปัญหา หรือ Algorithm ส่วนเรื่องอื่น ๆ เป็นเรื่องที่ผูกพันกับภาษาโปรแกรมที่เรียนอยู่นั้น
ผมจะพูดถึงเทคนิคการทำความเข้าใจ Algorithm วิธีศึกษา "ขั้นตอนวิธี" หรือ Algorithm นั้นจะต้องพยายามสร้างจินตนาการให้ได้ โดยเริ่มจะเรื่องง่าย ๆ ใกล้ ๆ ตัว

แรกสุดของนึกถึงการทำไข่เจียว ถ้าเราอยากกินไข่เจียวเราจะทำอย่างไร ลองหัดเขียนคำอธิบายขั้นตอนการทำตั้งแต่ต้นจนจบได้ไหมครับ แน่นอนครับผมคิดว่าทุกคนน่าจะทำไข่เจียวเป็น แต่ไม่แน่ว่าจะสามารถเขียนขั้นตอนการทำไข่เจียวได้ ถ้าคุณทำได้แสดงว่าใกล้ความเป็นนักโปรแกรมไปขั้นหนึ่งแล้ว แต่ถ้าทำไม่ได้ก็ควรไปหาตำราทำกับข้าวมาอ่าน ในนั้นอาจไม่มีวิธีทำไข่เจียว แต่ลองอ่านวิธีทำกับข้าวอย่างใดก็ได้ อ่านแล้วลองมานั่นเขียนวิธีทำไข่เจียวใหม่อีกทีหนึ่ง

ไม่เชื่อก็ต้องเชื่อแหละครับว่าตำราทำกับข้าวนี่แหละคือเบื้องต้นของการหัดเขียนโปรแกรม เมื่อรู้อย่างนี้ก็อย่าได้แปลกใจนะครับว่าเวลานี้นักเขียนโปรแกรมสตรีกำลังมีเพิ่มมากขึ้นเรื่อย ๆ ถัดจากไข่เจียวเราลองหัดเขียนขั้นตอนการทำงานอย่างอื่นดูบ้าง เช่น สมมติว่าเราต้องการไปตลาดหรือซูเปอร์มาร์เกตเพื่อเลือกซื้อของใช้สัก 10 อย่าง ลองเขียนเป็นขั้นตอนดูสิครับว่าเริ่มต้นตั้งแต่ออกจากบ้านจนกลับถึงบ้านเราทำอะไรบ้าง

ถัดจากเรื่องชีวิตประจำวันที่ไม่เกี่ยวกับคอมพิวเตอร์เลย ลองมาคิดขั้นตอนที่เกี่ยวกับตัวเลขดูบ้าง เช่น ถ้าหากมีตัวเลข 10 จำนวน ลองอธิบายวิธีคิดหน่อยสิครับว่าคุณจะหาเลขที่มีจำนวนมากที่สุดได้อย่างไรในชีวิตจริง เวลาเราเห็นเลข 10 จำนวน เช่น 12 , 72 , 90 , 15 , 7 , 4 , 38 , 78 , 45 , 61 เราบอกได้ทันทีว่าเลขที่ใหญ่ที่สุดคือ 90 แต่เราไม่สนใจคำตอบครับ เราสนใจว่าคุณใช้วิธีอย่างไรในการหาคำตอบ

จริง ๆ แล้วเราคงบอกวิธีทำไม่ได้ เพราะสมองของเราทำงานเร็วเกินไป พอมองเห็นแถวของตัวเลขก็หาคำตอบได้แล้ว ถ้าอย่างนั้นลองเปลี่ยนเป็นว่าในการหาเลขจำนวนที่ใหญ่ที่สุดนี้ เราจะได้เห็นเลขทีละจำนวน คือ สมมติว่ามีใครสักคนนั่งอยู่ข้างหน้าเราคอยชูป้ายหรือกระดานที่เขียนเลขเอาไว้ให้เราดูทีละจำนวน แรกสุดก็ชูเลข 12 แล้วก็มาก็ลบทิ้งเขียนเป็น 72 แล้วยกให้เราดูใหม่ ทำอย่างนี้เรื่อย ๆ ไปจนหมดจำนวนเลขทั้งหมด คราวนี้พอจะบอกได้ไหมครับว่าคิดอย่างไร หรือทำอย่างไรจึงได้คำตอบเป็นเลข 90 ตรงนี้ต้องลองลงมือเขียนเองละครับ วิธีนี้เป็นวิธีการคิดขั้นตอนเชิงแก้ปัญหา ซึ่งในชีวิตประจำวันเราต้องคิดบ่อย ๆ แต่ไม่เคยใส่ใจกับวิธีคิด ดังนั้นเราจึงไม่สามารถถ่ายทอดออกมาเป็นขั้นตอนได้ อีกวิธีหนึ่งที่น่าจะช่วยได้คือ การใช้ผังงาน หรือ Flowchart เข้าช่วยในการเรียบเรียงลำดับความคิด

พูดถึงเรื่องนี้หลายคนคงบอกว่าผมล้าสมัยมาก ไปขุดเอาของโบราณอะไรมาให้เรียนก็ไม่ทราบ แต่อย่าดูถูกนะครับ เป็นที่ทราบดีอยู่แล้วว่าสมองของเรามีสองซีกคือ ซีกขวากับซีกซ้ายซึ่งล้วนมีความเชี่ยวชาญต่างกัน สมองซีกซ้ายเชี่ยวชาญในเรื่องภาษา ตรรกะ ตัวเลข ลำดับงาน สัญลักษณ์ เหตุผล และรายละเอียด ส่วนสมองซีกขวาเชี่ยวชาญในด้าน ภาพ จังหวะ ภาษาจินตนาการ สีสัน รูปแบบ และอารมณ์

นักเขียนโปรแกรมเก่ง ๆ จึงต้องอาศัยสมองซีกซ้ายมาก เพราะเป็นเรื่องที่ตรงกับความถนัดพอดี แต่การใช้ความเชี่ยวชาญของสมองเพียงซีกเดียวอาจจะไม่เหมาะสมนัก เช่น สมองอาจล้าหรือเหนื่อยหน่าย เหมือนเวลาอ่านตำราหรือหนังสือที่มีแต่ตัวอักษร และตัวเลขเต็มพรืดไปหมดจะรู้สึกน่าเบื่อไม่อยากอ่านเลย ดีร้ายอาจอยากขว้างทิ้งไปด้วยซ้ำ แต่ถ้าเป็นหนังสือที่มีภาพประกอบสวย ๆ มีสีสัน หรือมีการ์ตูนประกอบละก็ ถึงจะเป็นหนังสือยาก ๆ ก็ยังอ่านได้ไม่เบื่อหรือรู้สึกล้าแต่อย่างใด

ผังงานหรือ Flowchart นั้นก็คือสัญลักษณ์ภาพที่ช่วยแสดงขั้นตอนการทำงาน ดังนั้นผังงานจึงช่วยกระตุ้นให้สมองซีกขวาทำงานไปพร้อม ๆ กับสมองซีกซ้าย ดังนั้นการคิดและความเข้าใจจึงดีขึ้นกว่าการเขียนขั้นตอนเป็นถ้อยคำหรือเป็นสูตรเป็นตัวเลขแต่เพียงอย่างเดียว

ดังนั้นอย่ามองข้ามผังงานเลยครับ เราควรมองว่าผังงานเป็นเครื่องมือสำคัญที่ช่วยให้เราคิดได้ดีขึ้นและสะดวกขึ้น ใครไม่ใช้ก็ช่างเขา แต่เรามาหัดใช้ให้เป็นจะเห็นประโยชน์มากทีเดียวครับ น่าเสียดายที่หนังสือเกี่ยวกับผังงานและอัลกอริธึมที่เขียนเป็นภาษาไทยไม่ค่อยมี หรือจะพูดว่าไม่มีก็

ไม่ผิด หนังสือ Algorithm ที่มีอยู่เวลานี้มักจะรวมเสนอเนื้อหาพร้อมกันไปกับเรื่องโครงสร้างข้อมูล (Data Structure) คือ กล่าวถึงขั้นตอนในการดำเนินการกับข้อมูลต่าง ๆ ที่จัดเก็บเป็นโครงสร้างรูปต้นไม้บ้าง เป็นรายการโยง (List) บ้าง จะหาที่เกี่ยวกับการแก้ปัญหาได้น้อยเต็มที

ถัดจาก Algorithm ก็เป็นแนวคิดหลักเกี่ยวกับการทำงานของคำสั่งและโปรแกรม ตรงนี้ยากสักนิดและไม่ค่อยมีใครกล่าวถึง แต่ผมคิดว่าผู้เรียนควรจะทราบสักนิดหน่อยว่าภาษานั้น ๆ มีหลักการอย่างไร ต้องสั่งอย่างละเอียดมากแค่ไหน สั่งลงไปถึงระดับบิตและไบต์ได้ไหม ภาษานั้น ๆ เหมาะกับงานประเภทใด เพราะเหตุใด ถ้าใช้ภาษาอื่นมาแทนจะเป็นอย่างใด จะดีกว่าหรือไม่

ภาพกว้าง ๆ เหล่านี้อาจจะเรียนเองได้ยาก เพราะไม่มีตำราภาษาไทยในด้านนี้อีกนั่นแหละครับ ทางที่ถูกอาจารย์นั่นแหละควรอธิบายในตอนแรก ไม่ต้องลงลึกมากนักเพราะนักเรียนยังไม่ทราบรายละเอียด อาจใช้วิธีสอนไปสอดแทรกความรู้เหล่านี้ไปก็ได้ ต่อมาคือ โครงสร้างของโปรแกรม ตรงนี้จะว่าไม่มีอะไรก็ได้ จะว่ามีให้เรียนและทำความเข้าใจให้มากก็ได้ ในที่นี้ผมไม่ได้หมายความว่าแต่ละโปรแกรมจะต้องแบ่งเป็น Division ต่าง ๆ อย่างในภาษาโคบอล แต่ผมหมายถึง Program Construct ที่บอกโครงสร้างที่เป็นไปได้ของคำสั่งและตัวโปรแกรม ยกตัวอย่างเช่น Construct ของภาษาโปรแกรมทั่วไปมีแบบ Sequence แบบ if - then - else และแบบ Loop แต่ Construct ในภาษา Prolog เป็นอีกแบบหนึ่ง หรือในภาษาสำหรับ Simulate เช่น IFPS นั้น ลำดับคำสั่งแบบ Sequence ไม่มีความหมาย

เรื่องนี้เรียนไม่ยากแต่เข้าใจยาก ผมคิดว่าในบางภาษาอาจจะต้องศึกษาก่อน อย่างภาษากลุ่ม Object Oriented ทั้งหลายไม่ว่าจะเป็น Smalltalk Object , Pascal หรือ C++ นั้น เราจำเป็นต้องทำความเข้าใจโครงสร้างเหล่านี้ก่อน แต่ถ้าเป็นภาษาง่าย ๆ เช่น เบสิก เราอาจลองมาศึกษาทีหลังก็ได้

คราวนี้ก็มาถึงเรื่องของรูปแบบคำสั่งซึ่งผู้สอนและสถาบันทุกแห่งเน้นเรื่องนี้ค่อนข้างมาก เพราะถือว่าโปรแกรมจะผิดหรือถูกขึ้นอยู่กับการเขียนคำสั่ง หากเขียนผิดไวยากรณ์รับประกันว่าโปรแกรมผิดแน่ ๆ แต่อาจารย์ลืมไปว่าถึงจะเขียนถูกโปรแกรมก็ยังผิดได้อยู่ดีนั่นแหละครับ เปรียบเสมือนพวกเราเขียนประโยคภาษาไทยที่ถูกต้องเรียงกันแปดประโยค อ่านแล้วถูกหมด แต่อาจจะไม่เป็นเรื่องเป็นราวหรือสื่อความหมายเลยก็ได้ ยิ่งจะให้เป็นบทกวีด้วยแล้วยิ่งไม่ต้องคิดฝัน

การเรียนรู้ภาษาโดยศึกษาทีละคำสั่งนั้น เป็นการมุ่งไปที่รายละเอียดเปรียบเสมือนเราเดินสำรวจต้นไม้ทีละต้น แต่ไม่รู้จักป่าที่ต้นไม้นั้นขึ้นอยู่ ทางที่ถูกเราต้องเริ่มจากภาพรวมคือ สภาพป่าก่อนแล้วจึงเข้าไปดูรายละเอียด

แนวคิดที่ให้ทำความเข้าใจภาษา คำสั่ง และโครงสร้างโปรแกรมก็เป็นแบบนี้คือ ต้องการให้รู้ภาพกว้าง ๆ ก่อนลงไปดูคำสั่ง ต้องการให้เกิดแนวคิดว่าจะเอาคอมพิวเตอร์ไปแก้ปัญหาอย่างไรก่อนจะลงมือเขียนคำสั่งเพื่อแก้ปัญหานั้น คราวนี้เมื่อถึงคราวต้องเรียนรู้แต่ละคำสั่งแล้ว เราต้องมีเคล็ดในการเรียนด้วยเหมือนกัน การเรียนรู้คำสั่งในภาษาคอมพิวเตอร์ให้เข้าใจ ไม่ได้อยู่ที่การรู้จักใช้คำสั่งให้ถูกต้องตลอดเวลา แต่รู้ว่าถ้าเขียนคำสั่งผิดจะเกิดอะไรขึ้น นักศึกษาของผมบอกผมอย่างยิ้มแย้มแจ่มใสว่า "ผมเขียนโปรแกรมไม่เคยผิดเลย" ผมตอบว่า "นั่นแหละคุณยังไม่ได้เรียนรู้มากนัก" วิธีที่ถูกต้องในการเรียนเรื่องเขียนคำสั่งคือ จะต้องทำผิดบ่อย ๆ แต่เป็นการผิดอย่างจงใจ ทำผิดเพื่อให้เกิดความเข้าใจว่าคอมพิวเตอร์หรือตัวแปลภาษาจะมีปฏิกิริยาต่อความผิดนั้นอย่างไร จะทำงานให้หรือไม่ ถ้าทำจะให้ผลลัพธ์อย่างไร

ยี่สิบปีมาแล้วผมคิดว่าผมเป็นยอดนักโปรแกรม ผู้รู้ภาษาฟอร์แทรนอย่างเจนจบ รู้คำสั่ง รู้โครงสร้าง เข้าใจการทำงานทุกอย่าง แต่วันที่เพื่อนรุ่นพี่ของผมที่มาใช้คอมพิวเตอร์ของผมถามคำถามง่าย ๆ ว่า "ถ้าโปรแกรม 1 โยงไปโปรแกรม 2 ได้ แล้วโปรแกรม 2 จะโยงกลับมายังโปรแกรม 1 ได้ไหม" ผลคือ ผมตอบไม่ได้เพราะไม่เคยฉุกคิดหรือทดลองเรื่องนี้ คำตอบของปัญหาแบบนี้อาจมีได้สองวิธี วิธีแรกคือ จากทฤษฎีภาษาโปรแกรม ส่วนวิธีที่สองคือ จากการทดสอบหรือทดลอง

ผมขอสนับสนุนให้ทดลองกับคำสั่งต่าง ๆ กับโครงสร้างแบบต่าง ๆ อย่าหยุดอยู่กับรูปแบบที่ถูกไวยากรณ์ ลองเปลี่ยนคำสั่งให้ผิดไปเล็ก ๆ น้อย ๆ แล้สังเกตผลลัพธ์ บันทึกผลลัพธ์ที่ได้อย่างเป็นระบบ โดยวิธีนี้แหละครับที่เราได้เรียนรู้ภาษาคอมพิวเตอร์แท้จริง จะขาดก็อีกเรื่องหนึ่งคือ การตรวจสอบการแก้ไขข้อผิดพลาดในโปรแกรม เรื่องนี้คงต้องว่ากันอีกยาว จึงขอยกยอดไปก่อน ถ้าไม่ลืมจะมาคุยเรื่องนี้กันใหม่

****