โปรแกรมการเข้ารหัสข้อความด้วยรูปภาพ

A PROGRAM FOR ENCRYPT DATA INTO DIGITAL IMAGE

ไกรฤกษ์ ทิมอ่อน
ประพนธ์ เอื้ออานันท์
โยธิน เทียนดี

ปัญหานี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต
ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์
คณะวิทยาศาสตร์
สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง
ปีการศึกษา 2546

A PROGRAM FOR ENCRYPT DATA INTO DIGITAL IMAGE

KRAIRURK TIM-ON

PRAPONH AUE-ANAN

YOTHIN TIANDEE

A SPECIAL PROJECT SUBMITTED IN PARTIAL FULFILLMENT OF THE REQUREMENT FOR THE DEGREE OF BACHELOR OF SCIENCE

DEPARTMENT OF MATHEMATICS AND COMPUTERR SCIENCE

FACULTY OF SCIENCE

KING MONGKUT'S INSTITUTE OF TECHNOLOGY LADKRABANG

ACADEMIC YEAR 2003

หัวข้อปัญหาพิเศษ	โปรแกรมการเข้ารหัสข้อความด้วยรูปภาพ		
	A PROGRAM FOR ENCRYPT DATA INTO DIGITAL IMAGE		
ชื่อนักศึกษา	นายไกรฤกษ์	ทิมอ่อน	43050005
	นายประพนธ์	เอื้ออานันท์	43050026
	นายโยธิน	เทียนดี	43050036
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์		
สาขาวิชา	คณิตศาสตร์ประยุกต์		
อาจารย์ที่ปรึกษา	รศ.ธีรวัฒน์ ประกอบผล		

ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ คณะวิทยาศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง อนุมัติให้นำปัญหาพิเศษนี้เป็นส่วนหนึ่งของการศึกษาตามหลักสูตรวิทยาศาสตรบัณฑิต สาขาวิชาคณิตศาสตร์ประยุกต์ ประจำปีการศึกษา 2546

คณะกรรมการสอบ		ลายมือชื่อ
ประธานกรรมการ	ผศ.ดร.ณัฐไชย ลีนาวงค์	
กรรมการ	อ.วรรณพร สรรประเสริฐ	
กรรมการและอาจารย์ที่ปรึกษา	รศ.ธีรวัฒน์ ประกอบผล	

(ผู้ช่วยศาสตราจารย์ ดร.วีระ บุญจริง)
หัวหน้าภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์

หัวข้อปัญหาพิเศษ	โปรแกรมการเข้ารหัสข้อความด้วยภาพ		
ชื่อนักศึกษา	นายไกรฤกษ์	ทิมอ่อน	43050005
	นายประพนธ์	เอื้ออานันท์	43050026
	นายโยธิน	เทียนดี	43050036
ปริญญา	วิทยาศาสตรบัณฑิต		
ภาควิชา	คณิตศาสตร์และวิทยาการคอมพิวเตอร์		
สาขาวิชา	คณิตศาสตร์ประยุกต์		
ปีการศึกษา	2546		
อาจารย์ที่ปรึกษา	รศ.ธีรวัฒน์ ประกอบผล		

บทคัดย่อ

การรักษาความลับของข้อมูลถือว่ามีความสำคัญมากในการสื่อสารข้อมูลกัน ระหว่างระบบเครือข่ายคอมพิวเตอร์ ดังนั้นในโครงงานพิเศษเรื่อง โปรแกรมการเข้ารหัสข้อความด้วยภาพ จึงพัฒนาขึ้นเพื่อเพิ่มความปลอดภัยในการสื่อสารซึ่งจะช่วยให้ข้อมูลมีความปลอดภัยมากขึ้นในการสื่อสารกันระหว่างระบบเครือข่ายคอมพิวเตอร์

โครงงานพิเศษนี้ได้ประยุกต์ใช้ความรู้ทางคณิตศาสตร์ เรื่องของความน่าจะเป็นในการหาค่าของตัวอักษรเพื่อใช้ในการลดขนาดข้อมูล นำความรู้ทางด้านการบีบอัดตัวอักษรแบบ Huffman มาใช้ในการลดขนาดข้อมูลที่จะซ่อนลงในรูปภาพและได้ศึกษาโครงสร้างของข้อมูลภาพแบบบิทแมพ (bitmap) และ GIF รวมทั้งวิธีการซ่อนข้อมูล (Data hiding) แบบ LSB (Least Significant Bit) ซึ่งอาศัยการซ่อนข้อมูลลงไปในบิทที่มีผลกระทบน้อยที่สุดของข้อมูลที่เป็นสื่อ การศึกษาดังกล่าวก็เพื่อที่จะนำข้อมูลซ่อนลงในส่วนข้อมูลสีของภาพ โดยที่ทำให้สีของภาพเปลี่ยนแปลงไปน้อยที่สุด โดยโครงงานนี้ได้ใช้ภาษา C ++ ของ Microsoft Visual C ++ 6.0 เป็นเครื่องมือสำคัญในการพัฒนา การทำงานของโปรแกรมจะแบ่งออกได้ 2 ส่วนใหญ่ ๆ คือ ส่วนการซ่อนข้อมูลลงในภาพและส่วนการดึงข้อมูลที่ซ่อนอยู่ในภาพ

Special Project Title	A PROGRAM FOR ENCRYPT DATA INTO DIGITAL IMAGE		
Students	Mr.Krairurk	Tim-on	43050005
	Mr.Praponh	Aue-anan	43050026
	Mr.Yothin	Tiandee	43050036
Degree	Bachelor of Science		
Department	Mathematics and Computer Science,Faculty of Science		
Programme	Applied Mathematics		
Academic Year	2003		
Special Project Advisor	Assoc.Prof.Teerawat Prakobphon		

ABSTRACT

The privacy control is an important thing in data communications. This special project studies the information hiding techniques. The purpose is to increase security for data communication in network

This special project applies mathematical knowledge to determining probability of occurency of character for compress data. Furthermore, Huffman coding compression is applied to reduce the size of information , structure of bitmap file and GIF file format and LSB (Least Significant Bit) data hiding technique has been studied and applied to make difference between before and after image imperceptible. In addition Microsoft Visual C++ version 6.0 is chosen as a main developing tool.The program consists of two main parts, which are the information hiding into image process and the display information from image.

กิตติกรรมประกาศ

ในการทำโครงงานปัญหาพิเศษเรื่องโปรแกรมซ่อนข้อมูลลงในรูปภาพเพื่อเพิ่มความปลอดภัยในการสื่อสารนี้ สามารถสำเร็จผ่านลุล่วงไปได้ด้วยดี ทางคณะผู้จัดทำต้องขอขอบพระคุณ รองศาสตราจารย์ ธีรวัฒน์ ประกอบผล อาจารย์ผู้รับผิดชอบโครงงานปัญหาพิเศษฉบับนี้ ที่กรุณาให้คำแนะนำและเป็นที่ปรึกษาในการแก้ปัญหาในด้านต่างๆรวมทั้งเป็นผู้ตรวจสอบความถูกต้องของปัญหาพิเศษฉบับนี้

ขอขอบพระคุณอาจารย์ทุกท่านที่ได้ประศาสน์วิชาความรู้ทั้งทางด้านทฤษฎีและภาคปฏิบัติแก่ทางคณะผู้จัดทำจนกระทั่งปัญหาพิเศษนี้สัมฤทธิ์ผลด้วยดีทุกประการ

ขอขอบพระคุณเจ้าหน้าที่ภาควิชาคณิตศาสตร์และวิทยาการคอมพิวเตอร์ที่คอยให้ความสะดวกในการใช้ห้องปฏิบัติการคอมพิวเตอร์

นอกจากนี้ทางคณะผู้จัดทำต้องขอขอบพระคุณเพื่อนๆ ทุกคนที่ให้ความช่วยเหลือในด้านต่างๆ ที่เกี่ยวข้องกับปัญหาพิเศษไว้ ณ ที่นี้

คณะผู้จัดทำ

มีนาคม 2547

สารบัญ

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญ (ต่อ)

หน้า

สารบัญตาราง

ตารางที่ **หน้า**

สารบัญรูป

รูปที่ **หน้า**

สารบัญรูป (ต่อ)

บทที่ 1

บทนำ (Introduction)

1.1 ความเป็นมาและความสำคัญของปัญหา

ในการสื่อสารข้อมูลกันของบุคคลสองฝ่ายนั้น เรื่องของความปลอดภัยของข้อมูลจะถือว่าเป็นเรื่องที่สำคัญมากเนื่องจากหากมีบุคคลที่สามได้ล่วงรู้ถึงข้อมูลนั้นๆ อาจก่อให้เกิดความเสียหายได้ ดังนั้นมนุษย์จึงได้คิดวิธีการนำข้อมูลที่จะสื่อสารกันไปผ่านระบวนการหนึ่งซึ่งทำให้ข้อมูลนั้นเข้าใจกันเฉพาะบุคคลสองฝ่ายเท่านั้น เมื่อบุคคลที่สามได้รับข้อมูลนั้นไปจะไม่สามารถเข้าใจได้โดยง่ายต่อมามนุษย์ได้มีความรู้ด้านเทคโนโลยีต่างๆเพิ่มมากขึ้นจึงได้ประยุกต์วิธีที่จะเพิ่มความปลอดภัยของข้อมูลที่จะสื่อสารกัน ให้มีวิธีที่หลากหลายมากขึ้น หนึ่งในนั้นก็ คือวิธีการซ่อนข้อมูลที่จะสื่อสารกันลงในข้อมูลชนิดอื่น เช่น ซ่อนข้อมูลที่จะสื่อสารกันลงไปในภาพ หรือ ซ่อนข้อมูลที่จะสื่อสารกันลงไปในข้อมูลเสียง ฯลฯ ในตัวโครงงานปัญหาพิเศษ ที่จะได้ศึกษาต่อไปนี้ จะเป็นการเพิ่มความปลอดภัยโดยซ่อนข้อมูลที่จะสื่อสารกันลงไปในรูปภาพ โดยก่อนการซ่อนข้อมูลจะทำการบีบอัดข้อมูลนั้นๆ ก่อนเพื่อให้มีขนาดเล็กลงก่อนที่จะทำการซ่อน

1.2 ความมุ่งหมายและวัตถุประสงค์ของการศึกษา

1. เพื่อเพิ่มความปลอดภัยของข้อมูลในการสื่อสารกันระหว่างบุคคลสองฝ่าย
2. เพื่อมีความรู้ความเข้าใจในเรื่องโครงสร้างของไฟล์รูปภาพ ที่จะนำมาใช้ในโครงงานปัญหาพิเศษนี้
3. เพื่อมีความรู้ความเข้าใจในกระบวนการบีบอัดข้อมูลชนิดที่จะนำมาใช้ในโครงงานปัญหาพิเศษนี้
4. เพื่อเป็นแนวทางให้นำไปพัฒนาต่อโดยการซ่อนข้อมูลที่ต้องการจะสื่อสาร ลงไปในสื่อชนิดอื่นๆ เช่น เสียง ฯลฯ

1.3 ขอบเขตของการศึกษา

โปรแกรมนี้มีหน้าที่ซ่อนข้อความลงไปในภาพ โดยภาพที่จะถูกซ่อนข้อความลงไปนั้นจะมีการเปลี่ยนแปลงเกิดขึ้นน้อยที่สุด จนตาของมนุษย์ไม่สามารถที่จะสังเกตการเปลี่ยนแปลงนั้นได้ ซึ่งขอบเขตของโครงงานนี้คือ

1. ภาพที่จะนำมาใช้กับโปรแกรมนี้จะใช้ได้เฉพาะภาพชนิด Bitmap และ GIF เท่านั้น
2. ภาพที่ใช้จะได้มาจากผู้ใช้ซึ่งโปรแกรมไม่สามารถสร้างภาพนี้ขึ้นมาเองได้

3. ข้อความจะรับมาจากผู้ใช้โดยตรงหรือเปิดอ่าน Text File(.txt)

4. ศึกษาขั้นตอนการลดขนาดข้อมูล เพื่อประโยชน์ในการเพิ่มปริมาณข้อความที่ซ่อน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

การทำโครงงานปัญหาพิเศษนี้คาดว่าจะได้รับประโยชน์ ดังนี้

1. สามารถนำโปรแกรมนี้มาใช้ในการเพิ่มระดับความปลอดภัยของข้อมูลได้

2. เพื่อเป็นแนวทางในการศึกษาต่อในเรื่องการซ่อนข้อความด้วยภาพ หรืออาจเป็นแนวทางในการพัฒนาต่อโดยซ่อนข้อความลงในสื่อชนิดอื่นๆได้

3. มีความรู้ความเข้าใจในรูปแบบโครงสร้างของภาพ Bitmap และ GIF ให้เพิ่มมากขึ้น และหวังว่าจะเป็นประโยชน์ต่อผู้ที่ต้องการหาความรู้ที่เกี่ยวข้องกับโครงสร้างภาพ Bitmap และ GIF ได้ในอนาคต

4. มีความรู้ความเข้าในกระบวนการบีบอัดข้อมูลเพิ่มมากขึ้น

5. มีความรู้ความเข้าใจในทฤษฎี Steganography เพิ่มมากขึ้น

1.5 ขั้นตอนการศึกษา

การศึกษาเพื่อทำปัญหาพิเศษได้แบ่งออกเป็น ส่วนดังนี้

1. ศึกษาถึงส่วนประกอบของโครงสร้างภาพ Bitmapเพื่อที่จะได้ทราบว่าการซ่อนข้อมูลควรที่จะซ่อนไว้ที่ส่วนใดของภาพ เพื่อที่จะทำให้ภาพนั้นเกิดการเปลี่ยนแปลงที่น้อยที่สุดและขนาดของภาพต้องไม่เปลี่ยนแปลง

2. ศึกษาถึงส่วนประกอบของโครงสร้างภาพ GIF เพื่อที่จะได้ทราบว่าหากต้องการที่จะซ่อนข้อมูลควรจะซ่อนไว้ที่ส่วนใดของภาพ เพื่อที่จะทำให้ภาพนั้นเกิดการเปลี่ยนแปลงให้น้อยที่สุด

3. ศึกษาการบีบอัดข้อมูลโดยใช้กระบวนการ Huffman Coding เพื่อนำมาประยุกต์ใช้ในการทำโครงงานปัญหาพิเศษนี้

4. ศึกษาถึงการบีบอัดข้อมูลชนิด LZW เนื่องจากภาพ GIF ใช้การบีบอัดข้อมูลในรูปลักษณะนี้

5. ศึกษาการใช้โปรแกรม Microsoft Visual C++ เพื่อใช้ในการพัฒนาโปรแกรมในโครงงานปัญหาพิเศษนี้ ให้ตรงตามความต้องการมากที่สุด

6. ศึกษาการใช้งาน HEX Editor เพื่อใช้ในการดูโครงสร้างของไฟล์ภาพ

7. วิเคราะห์และออกแบบการทำงานของโปรแกรมโดยการเอาวิธีการทางคอมพิวเตอร์ที่ได้ศึกษามา เข้ามาช่วยในการวิเคราะห์และออกแบบขั้นตอนและโครงสร้างของโปรแกรมโดยจะทำการแบ่งงานออกเป็นส่วนๆ เช่น ส่วนเลือกภาพที่จะซ่อน ส่วนรับข้อความ ส่วนซ่อนข้อความ โดยการทำงานในภาพชนิด Bitmap กับ GIF จะมีการทำงานที่แตกต่างกัน

8. ทำการพัฒนาโปรแกรมตามที่ได้ออกแบบเอาไว้ในขั้นตอนของการวิเคราะห์ และออกแบบการทำงานของโปรแกรม

9. ทำการทดสอบโปรแกรม และปรับปรุงการทำงานของโปรแกรม ให้มีประสิทธิภาพตามที่ต้องการ และบอกถึงความสามารถทั้งหมด รวมถึงข้อจำกัดต่างๆของโปรแกรม

10. ทำเอกสารประกอบการใช้งานโปรแกรมและเอกสารแสดงส่วนอ้างอิงต่างๆ ที่ใช้ในการศึกษาและพัฒนาโปรแกรม

บทที่ 2
ทฤษฎีที่เกี่ยวข้อง

ปัญหาพิเศษนี้มีแนวคิดและทฤษฎีที่เกี่ยวข้องคือ โครงสร้างของภาพบิทแมพ (Bitmap) , โครงสร้างไฟล์ภาพ GIF ทฤษฎีความน่าจะเป็นในการลดขนาดข้อมูล และทฤษฎี Steganography ซึ่งเป็นทฤษฎีที่ว่าด้วยการซ่อนข้อมูลลงในสื่อต่างๆ

2.1 ภาพ 24- bit bitmap

จากโครงสร้างของภาพ bitmap Header ของภาพจะอยู่ในไบต์ที่ 1 ถึง ไบต์ที่ 53 ซึ่ง Header นี้เป็นส่วนที่อธิบายคุณสมบัติของภาพ bitmap แต่ละภาพ โดยที่ Header ของภาพแต่ละภาพ bitmap มีค่าไม่เหมือนกัน ดังนั้น ในการที่จะซ่อนข้อความลงในรูปภาพ จึงต้องเริ่มต้นซ่อนลงไปในไบต์ที่ 54 เป็นต้นไป ซึ่งเป็นส่วนของ image data

2.2 ภาพ GIF

จากโครงสร้างของภาพ GIF ทำให้ทราบว่าข้อมูลของภาพที่อยู่ในส่วนของ Raster Data มีการจัดเก็บ โดยการผ่านการเข้ารหัสแบบ LZW ดังนั้นการที่จะซ่อนข้อมูลลงในภาพ จึงต้องทำการถอดรหัสข้อมูลของภาพออกมาก่อน แล้วค่อยทำการซ่อนข้อความเมื่อซ่อนข้อความเสร็จแล้วจึงค่อยนำข้อมูลใหม่ที่ได้ไปเข้ารหัสแบบ LZW อีกครั้ง แล้วจึงค่อยนำไปเก็บไว้ในส่วนของ Raster Data ดังเดิม

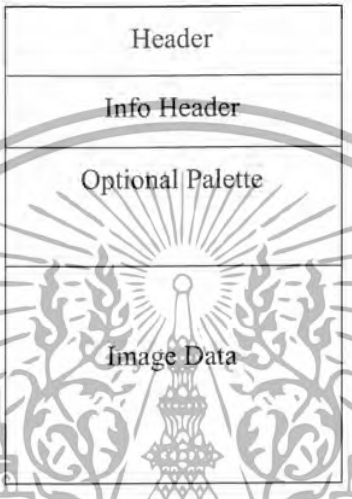
2.3 โครงสร้างไฟล์รูปภาพ Bitmap (Bitmap image format)

ไฟล์ภาพ Bitmap จะมีนามสกุลเป็น .BMP หรือบางครั้งจะใช้นามสกุลเป็น .DIB ซึ่งไฟล์ Bitmap นี้จะถูกจัดเก็บอยู่ในรูปแบบของ device – independent bitmap (DIB) ซึ่งการจัดการดังกล่าวจะทำให้ Windows สามารถที่จะแสดงภาพ 24 – bits bitmap ดังกล่าวบน display device แบบใดๆก็ได้ โดยคำว่า "device independent" หมายความว่า bitmap จะระบุ pixel color ในรูปแบบที่ไม่ขึ้นกับวิธีที่ถูกใช้ในการแสดงสี

2.3.1 Bitmap-File Structures

โครงสร้างของ bitmap-file ประกอบด้วย

- a bitmap-file header
- a bitmap-information header
- an optional palette
- an image data



รูปที่ 2.1 แสดงโครงสร้างของ Bitmap File

โดยไฟล์ bitmap จะอยู่ในรูปแบบดังต่อไปนี้

```
BITMAPFILEHEADER    bmfh;
BITMAPINFOHEADER    bmih;
RGBQUAD             aColor[];
BYTE                aBitmapBits[];
```

BITMAPFILEHEADER

Bitmap-file header จะเป็นส่วนของข้อมูลที่เกี่ยวกับ type , size และ layout of a divice-independent bitmap file โดย bitmap- file header จะถูกนิยามในรูปของ BITMAPFILEHEADER structure ในที่นี้กำหนด UINT มีขนาด 2 byte , DWORD มีขนาด 4 byte

```
Typedef struct tagBITMAPFILEHEADER{/*bmfh*/}
        UINT        bfType;
        DWORD       bfSize;
        UINT        bfReserved1;
        UINT        bfReserved2;
        DWORD   bfOffBits;
```

ตารางที่ 2.1 อธิบายโครงสร้าง BITMAPHEADER

ไบต์ที่	ขนาด	ชื่อตัวแปร/โครงสร้าง	ค่ามาตราฐาน	อธิบาย
1	2	BfType	19778	ระบุชนิดของ file ซึ่งจะต้องมีค่าเป็น 'BM' ซึ่งค่านี้จะเป็นค่าที่ระบุว่าไฟล์นี้ไป .bmp ไฟล์
3	4	BfZize	ไม่แน่นอน	ระบุขนาดของไฟล์ซึ่งมีหน่วยเป็นไบต์
7	2	BfReserved1	0	สงวนไว้ซึ่งต้องมีค่าเป็น 0
9	2	BfReserved2	0	สงวนไว้ซึ่งต้องมีค่าเป็น 0
11	4	bfOffBits	1078	ตั้งแต่ต้นไฟล์จนถึงข้อมูลภาพ (bitmap data) ว่ามีขนาดกี่ไบต์

BITMAPINFOHEADER

```
Typedef struct tagBITMAPINFOHEADER{/*bmih*/}
        DWORD   biSize;
        LONG    biWidth;
        LONG    biHeight;
        WORD    biPlanes;
        WORD    biBitCount;
        DWORD   biCompression;
        DWORD   biSizeImage;
        LONG    biXPelsPerMeter;
        LONG    biYPelsPerMeter;
        DWORD   biClrUsed;
```

DWORD biClrImportant;

ตารางที่ 2.2 อธิบายโครงสร้าง BITMAPINFOHEADER

ไบต์ที่	ขนาด	ชื่อตัวแปรโครงสร้าง	ค่ามาตราฐาน	อธิบาย
15	4	BiSiZe	40	ระบุขนาดที่ใช้ในส่วนของ BITMAPINFOHEADER มีหน่วยเป็นไบต์
19	4	BiWidth	ตามขนาดภาพ	ระบุความกว้างของรูปมีหน่วยเป็น pixel
23	4	BiHeight	ตามขนาดภาพ	ระบุความสูงของรูปมีหน่วยเป็น pixel
27	2	BiPlanes	1	ระบุจำนวน planes สำหรับ tangent device โดยต้องมีค่าเป็น 1
29	2	BiBitCount	ตามรูปแบบของภาพ	ระบุจำนวนบิตที่ใช้ใน 1 pixel โดยถ้ามีค่า **1** – จะเป็น Bitmap แบบ Monochrome และ color table จะมี 2 entries โดยทุกๆ bit ในbitmap array จะแทนแต่ละ pixel **4** – Bitmapจะมี maximum color เป็น 16 สี แต่ละ pixel ใน bitmap จะถูกแทนโดยข้อมูล 4-bit ที่ชี้ไปยัง color table **8** – Bitmapจะมี maximum color เป็น 256 สี แต่ละ pixel ใน bitmap จะถูกแทนโดยข้อมูล 8-bit(1byte) ที่ชี้ไปยัง color table **24** - Bitmapจะมี maximum color เป็น 2^{24} และทุกๆ 3-byte sequence ใน bitmap array จะแทนความเข้ม

ไบต์ที่	ขนาด	ชื่อตัวแปรโครงสร้าง	ค่ามาตราฐาน	อธิบาย
				สัมพัทธ์ของสีแดง สีเขียว และสีน้ำเงินของแต่ละ pixel
31	4	biCompression	ตามรูปแบบการบีบอัด	ระบุประเภทของการบีบอัดข้อมูล โดยถ้ามีค่า **0** - ระบุว่าภาพ bitmap นั้นไม่มีการบีบอัด 1 – ระบุว่า มีการใช้ run-tenght encoded format สำหรับภาพ bitmap แบบ 8-bits per pixel 2 - ระบุว่า มีการใช้ run-tenght encoded format สำหรับภาพ bitmap แบบ 4-bits per pixel
35	4	biSizeImage	0	ระบุขนาดของภาพโดยมีหน่วยเป็นไบต์ ซึ่งถ้า biCompression มีค่าเท่ากับ 0 สามารถที่จะกำหนดให้ค่า biSizeImage นี้มีค่าเท่ากับ 0 ได้
39	4	biXPelsPerMeter	0	ระบุ Horizontal resolution ของ target ในหน่วย pixel per meter
43	4	biYPelsPerMeter	0	ระบุ Vertica resolution ของ target device ในหน่วย pixel per meter
47	4	biClrUsed	0	ระบุจำนวน color index ที่ถูกใช้จริงๆในภาพ bitmap ซึ่งค่าเป็น 0 จะใช้จำนวนสีเท่ากับจำนวนสีสูงสุด ที่ระบุโดยตัวแปร biBitCount
51	4	biClrImportant	0	ระบุจำนวนของสีที่มีความสำคัญสำหรับ bitmap โดยค่านี้มีค่าเท่ากับ 0 จะหมายความว่าทุกๆสีมีความสำคัญเท่ากันหมด

RGBQUAD

Color table จะถูกนิยามในรูปของ array of RGBQUAD structure ซึ่งจะเป็นส่วนที่จัดเก็บ element ของสีทั้งหมดที่ใช้ใน bitmap อย่างไรก็ตามในภาพ bitmap แบบ 24 bits จะไม่มีการใช้ตารางสี เพราะว่าทุกๆ pixel ในภาพ จะแทนด้วย 2-bits-red-green-blue (RGB) เลย จึงไม่ต้องใช้ตารางสี color ที่เก็บอยู่ใน color table ควรจะจัดเก็บโดยเรียงตามลำดับความสำคัญ เพราะว่าจะช่วยให้ display driver สามารถประมวลผลภาพได้ในกรณีที่ device นั้น ๆ ไม่สามารถแสดงสีได้เท่ากับจำนวนสีในภาพ bitmap ที่ต้องการเปิด ซึ่งถ้าเป็น Windows DIB version 3.0 หรือใน version ต่อ ๆ มา device driver จะสามารถใช้ biClrImportant ใน BITMAPFIEHEADER ในการตัดสินใจว่า colors ไหนมีความสำคัญ

```
Typedef struct tagRGBQUAD {/*rgbq*/}
        BYTE    rgbBlue;
        BYTE    rgbGreen;
        BYTE    rgbRed;
        BYTE    rgbReserved;
RGBQUAD;
```

ตารางที่ 2.3 อธิบายโครงสร้าง RGBQUAD array

ไบต์ที่	ขนาด	ชื่อตัวแปรโครงสร้าง	ค่ามาตรฐาน	อธิบาย
1	1	rgbBlue	-	ระบุความเข้มของสีน้ำเงิน
2	1	rgbGreen	-	ระบุความเข้มของสีเขียว
3	1	rgbRed	-	ระบุความเข้มของสีแดง
4	1	rgbReserved	-	ค่าในส่วนนี้ต้องกำหนดให้มีค่าเป็น0

2.3.2 Bitmap Compression

Windows Bitmap version 3.0 และ version ต่อ ๆ มาจะสนับสนุน run-length encoded (RLE) formats เพื่อใช้สำหรับในการบีบอัด bitmap ที่เป็นแบบ 4-bits per pixel และแบบ 8-bits per pixel ซึ่งการบีบอัดจะทำให้ลดการใช้ disk และ memory storage

2.3.2.1 Compression of 8-Bits-Pixel Bitmaps

ถ้าค่า biCompression ใน BITMAPINFOHEADER มีค่าเป็น 1 แสดงว่ามีการบีบอัดโดยใช้ run-length encoded format สำหรับภาพ bitmap แบบ 256 สี โดยใน format นี้จะมีอยู่ 2 โหมด คือ encoded mode และ absolute mode ซึ่งทั้ง 2 โหมดดังกล่าวจะปรากฏในส่วนใดของภาพ bitmap หนึ่งๆก็ได้

encoded mode

จะใช้ข้อมูลทีละ 2 byte โดย byte แรกจะระบุจำนวน consecutive pixel ที่จะถูกวาดโดยใช้ color index ใน byte ที่สอง อย่างไรก็ตามใน byte ที่หนึ่งนั้นสามารถมีค่าเป็น 0x00 ซึ่งระบุว่าเป็นตัว escape โดยค่าใน byte ที่สองที่ตามมาจะต้องมีค่าอยู่ระหว่าง 0x00 และ 0x02 โดยถ้ามีค่าเป็น 0x00 จะหมายถึง delta นั่นคือ อีก 2 byte ที่ตามมาจะเป็นค่าที่จะระบุ horizontal และ vertical offset จากตำแหน่ง ของ pixel ถัดไปที่จะวาด

absolute mode

จะใช้ข้อมูลทีละ 2 byte โดย byte แรกจะต้องมีค่าเป็น 0x00 และใน byte ที่สองที่ตามมาจะต้องมีค่าอยู่ระหว่าง 0x03 และ 0xFF โดยตัวเลขใน byte ที่สองนี้จะบอกถึงจำนวนของ byte ที่ตามมาซึ่งค่าของแต่ละ byte จะเป็น color index ของ 1 pixel ที่จะวาดโดยที่จำนวนข้อมูล ในส่วนนี้จะต้องมีจำนวนที่เป็นจำนวนเท่าของ word boundary

ตัวอย่าง 2.3.2.1.1 ของ 8-bit RLE bitmap (โดยค่าตัวเลขฐาน 16 สองหลัก ที่อยู่ในคอลัมน์ที่สองจะเป็น color index สำหรับ 1 pixel)

Compressed data	*Expanded data*
03 04	04 04 04
05 06	06 06 06 06 06
Compressed data	*Expanded data*
00 03 45 56 67 00	45 56 67
02 78	78 78
00 02 05 01	Moves 5 right and 1 down
02 78	78 78
00 00	End of line
09 1E	1E 1E 1E 1E 1E 1E 1E 1E 1E
00 01	End of RLE bitmap

2.3.2.2 Compression of 4-Bits-per Pixel- Bitmaps

ถ้าค่า biCompression ใน BITMAPINFOHEADER มีค่าเป็น 2 แสดงว่ามีการบีบอัดโดยใช้ run-length encoded format สำหรับภาพ bitmap แบบ 16 สี โดยใน format นี้จะมีอยู่ 2 โหมด คือ encoded mode และ absolute mode ซึ่งทั้ง 2 โหมดดังกล่าว จะปรากฏในส่วนใดของภาพ bitmap หนึ่งๆก็ได้

encoded mode

จะใช้ข้อมูลทีละ 2 byte โดย byte แรก จะระบุจำนวน pixel ที่จะถูกวาดโดยใช้ color index ใน byte ที่สอง ซึ่งใน byte ที่สองจะมี color index อยู่ 2 index โดย color index แรก จะอยู่ที่ high-nibble ส่วนอีก color index จะอยู่ที่ low-order nibble

pixel แรกจะถูกวาดโดยใช้สีที่มี color index อยู่ในส่วนของ high-order nibble , pixel ที่สองจะถูกวาดโดยใช้สีที่มี color index อยู่ในส่วนของ low-order nibble , pixel ที่สามจะถูกวาดโดยใช้สีที่มี color index อยู่ในส่วนของ high-order nibble เป็นเช่นนี้เรื่อยไปจนกระทั่งจำนวน pixel ที่ถูกวาดครบตามที่ระบุไว้ในข้อมูล byte แรก

อย่างไรก็ตามใน byte ที่หนึ่งนั้นสามารถมีค่าเป็น 0x00 ซึ่งระบุว่าเป็นตัว escape โดยค่าใน byte ที่สองที่ตามมาจะต้องมีค่าอยู่ระหว่าง 0x00 และ 0x02 โดยถ้ามีค่าเป็น 0x00 จะหมายถึง delta นั่นคือ อีก 2 byte ที่ตามมาจะเป็นค่าที่จะระบุ horizontal และ vertical offset จากตำแหน่งของ pixel ถัดไปที่จะวาด

absolute mode

จะใช้ข้อมูลทีละ 2 byte โดย byte แรก จะต้องมีค่าเป็น 0x00 และใน byte ที่สองที่ตามมาจะต้องมีค่าอยู่ระหว่าง 0x03 และ 0xFF โดยตัวเลขใน byte ที่สองนี้จะบอกถึงจำนวนของ color index ที่ตามมา ซึ่งค่าของแต่ละ byte ที่ตามมาจะมี 2 color index คือ high-nibble และ low-nibble โดยที่ 1color index ก็จะสำหรับ 1 pixel และจำนวนข้อมูล ในส่วนนี้จะต้องมีจำนวนที่เป็นจำนวนเท่าของ word boundary

ตัวอย่าง 2.3.2.2.1 ของ 4 –bit RLE bitmap(โดยค่าตัวเลขฐาน16 หนึ่งหลักที่อยู่ในคอลัมน์ที่สองจะเป็น color index สำหรับ 1 pixel)

Compressed data	*Expanded data*
03 04	0 4 0
05 06	0 6 0 6 0
00 03 45 56 67 00	4 5 5 6 6 7
02 78	7 8 7 8
00 02 05 01	Moves 5 right and 1 down
02 78	7 8 7 8
00 00	End of line
09 1E	1 E 1 E 1 E 1 E 1
00 01	End of RLE bitmap

2.4 โครงสร้างรูปภาพ GIF (Graphics Interchange Format)

โครงสร้างของ GIF ประกอบด้วย

รูปที่2.2 แสดงโครงสร้างภาพของ GIF

GIF Signature

GIF Signature เป็นส่วนที่แสดงว่านี้เป็นรูปภาพ GIF ที่สมบูรณ์. มันประกอบด้วยตัวอักษรที่เป็นค่าที่แน่นนอน 6 ตัวดังต่อไปนี้:

G I F 8 7 a

โดยที่ตัวอักษร 3 ตัวสุดท้าย 87a อาจจะถูกดูเป็นหมายเลขเวอร์ชันที่ใช้ สำหรับภาพ GIF ที่เจาะจงนี้และจะถูกใช้ในเอกสารอ้างอิงทั่วไปเกี่ยวกับภาพ GIF

Sceen Descriptor

Sceen Descriptor จะอ้างถึงตัวแปรครอบคลุมสำหรับรูปภาพ GIF ทั้งหมด มันกำหนดมิติครอบคลุมของช่องว่างรูปภาพ หรือ logical screen ที่ต้องการ, การมีอยู่จริงของข้อมูลที่ตารางสี, ข้อมูลความลึกสี, และสีจอพื้นหลัง ข้อมูลนี้ถูกตั้งอยู่ในชุดของไบท์ 8-บิท

Global Color Map

Global Color Map คือข้อกำหนดเพิ่มเติม แต่แนะนำสำหรับรูปภาพที่ซึ่ง มีสีแน่นอน การมีอยู่ของตารางสีนี้ จะถูกแสดงเอาไว้ว่า ถ้าสถานะ 'M' ของไบท์ 5 ของ Sceen Descriptor ถูกเซต เป็น 1 ก็จะมีตารางสีย่อยนี้ขึ้นมาให้ใช้ใน แฟ้มภาพ GIF

Image Descriptor

Image Descriptor จะกำหนดการบรรจุแต่งตั้งตามความเป็นจริงและขอบเขตของรูปภาพภายในช่องว่างที่กำหนดใน Image Descriptor และยังกำหนดค่าเพื่อแสดงสีภายใน ตารางสีย่อย , และเพื่อกำหนด อันดับการแสดงของแต่ละ pixel.

Local Color Map

Local Color Map คือข้อกำหนดเพิ่มเติมและการกำหนดที่นี่สำหรับใช้ในอนาคต ถ้า 'M' บิทของไบท์ที่10 ของ Image Descriptor ถูกเซต , แล้วตารางสีย่อย ที่ตาม Image Descriptor จะตอบรับกับส่วนรูปภาพที่ตามมานี้เท่านั้น ที่ตอนจบของรูปภาพ, ตารางสีจะย้อนกลับไปยังที่ที่กำหนดหลัง Screen Descriptor บันทึกว่า 'pixel' ของไบท์ ที่ 10 ของ Screen Descriptor จะถูกใช้ถ้า Local Color Map ถูกแสดง.

Raster Data

รูปแบบจริงๆของภาพ จะถูกกำหนดโดยลำดับของ index color (เลขชี้ตารางสี) โดยการเก็บของ pixel นี้จะเก็บจากซ้ายไปขวาและบนลงมาล่าง แต่ถ้า Bit "I" ถูกกำหนดให้เป็น 1 การจัดเรียงข้อมูล index สีมันจะเรียง แบบ วิธีการผ่าน 4 ขั้น (4-pass process) ในที่ซึ่งรูปภาพจะถูกใส่ลงไปข้างใน

- เขียนผ่านทุกๆแถวที่ 8 โดยเริ่มต้นที่แถวบนสุดรูปภาพ
- เขียนผ่านทุกๆ แถวที่ 8 โดยเริ่มต้นที่แถวที่ 5 จากข้างบน
- เขียนผ่านทุกๆแถวที่ 4 โดยเริ่มต้นที่แถวที่ 3 จากแถวบนสุด
- เสร็จสิ้นรูปภาพ, เขียนผ่านทุกแถวอื่นๆ, โดยเริ่มต้นที่แถวที่ 2 จากแถวบนสุด

GIF Terminater

ใช้สำหรับ การยกเลิกของแฟ้มรูปภาพ GIF , ตัวถอดรหัส GIF จะประมวลผลตอนจบของโหมด GIF เมื่อ ตัวอักษร 0x3B hex หรือถูกค้นพบ ดยการทำงานซอฟแวร์ที่ถอดรหัสจะค้างและจะรอการกระทำการแสดงว่าผู้ใช้พร้อมเพื่อดำเนินงานต่อไป.

GIF Extension Blocks

เป็นส่วนที่เตรียมไว้สำหรับ การขยายนิยามของภาพ GIF, สำหรับการกำหนดรูปแบบของการขยายของข้อมูลภายในภาพ GIF ใช้เพิ่มเติมอธิบาย หากเกิดปัญหาต่างๆ เช่น ถ้าภาพที่ต้องการจะเปิดเป็นภาพเคลื่อนไหว ส่วนนี้จะบอกว่าต้องใช้อะไรในการเปิดเป็น

ตารางที่ 2.4 อธิบายโครงสร้าง Basic File Format

ชื่อ	ขนาด	รายละเอียด
Signature	6 bytes	ค่ามาตรฐานของ GIF มีค่าเป็น 'GIF87a' หรือ 'GIF89a'
GlobalDescriptor	7 bytes	อธิบายคุณสมบัติโดยรวมของภาพ
Width	2 bytes	ความกว้างของภาพมีหน่วยเป็น pixel
Height	2 bytes	ความสูงของภาพมีหน่วยเป็น pixel
Flags	1 byte	อธิบายสถานะ
GlobalColorMap	bit 7	=1 ถ้าใช้ตารางสีหลัก =0 ถ้าใช้ตารางสีของแต่ละภาพ
ColorResolutionBits	bits 6-4	+1 = จำนวนบิทต่างสีในตารางสีหลัก

ชื่อ	ขนาด	รายละเอียด
reserved	bit 3	สงวนไว้ โดยปกติมีค่าเท่ากับ 0
PixelBits	bits 2-0	จำนวนของตารางสีหลัก = $2^{Pixel+1}$
BackgroundColor	1 byte	เลขดรรชนีของสีพื้นหลัง
AspectRatio	1 byte	เก็บค่าอัตราส่วน โดยปกติ = 0
GlobalColorMap	NumberOfGlobalColors * 3	ตารางสีหลัก ซึ่งจะมีต่อเมื่อ Flags.GlobalColorMap = 1
Red	1 byte	ระบุความเข้มของสีแดง
Green	1 byte	ระบุความเข้มของสีเขียว
Blue	1 byte	ระบุความเข้มของสีน้ำเงิน
repeated NumberOfGlobalColors times		
ExtensionBlocks	cannot be pre calculated	ตัวเลือกนี้อาจแสดงค่าได้หลายค่าในแต่ละครั้ง เราจะรู้ได้โดยการอ่านค่าใน Function code
Header	1 byte	มีค่ามาตรฐาน = 21H
FunctionCode	1 byte	เก็บค่าของ function code โดยจะอธิบายต่อไปในตารางของ function code
Length	1 byte	เก็บค่าความยาวของชุดข้อมูลที่ตามมามีหน่วยเป็นไบต์
Data	Length bytes	เก็บข้อมูล ดังแสดงในตารางของ function code
repeated any number of times. read first byte to check for terminator		
Terminator	1 byte	ตัวเลขแสดงการจบของข้อมูล มีค่า = 0
LocalDescriptor	10 bytes	อธิบายคุณสมบัติเฉพาะของแต่ละภาพ
Header	1 bytes	มีค่ามาตรฐาน = 2C
PosX	2 bytes	ค่าตำแหน่งของแนวแกน x
PosY	2 bytes	ค่าตำแหน่งของแนวแกน y
Width	2 bytes	ความกว้างของภาพ
Height	2 bytes	ความยาวของภาพ

ชื่อ	ขนาด	รายละเอียด
Flags	1 byte	อธิบายลักษณะเฉพาะ
LocalColorMap	bit 7	=1 ถ้าใช้ตารางสีของภาพ =0 ใช้ตารางสีหลัก
InterlacedImage	bit 6	=1 มีการเรียงแบบ Interlaced ! =0 ไม่มีการเรียงแบบ Interlaced
Sorted	bit 5	โดยปกติ = 0
reserved	bit 4-3	มีค่า เป็น 0 สงวนไว้
PixelBits	bit 2-0	จำนวนของตารางสีรอง = $2^{Pixel+1}$
LocalColorMap	NumberOfLocalColors * 3	ตารางสีรองซึ่งจะมีก็ต่อเมื่อ Flags.LocalColorMap = 1
Red	1 byte	ระบุความเข้มของสีแดง
Green	1 byte	ระบุความเข้มของสีเขียว
Blue	1 byte	ระบุความเข้มของสีน้ำเงิน
repeated NumberOfLocalColors times		
Raster Data Block	cannot be pre calculated	เก็บข้อมูลของรูปภาพ
InitialCodeSize	1 byte	จำนวนบิทต่อ 1 ค่าสี
Length	1 byte	เก็บค่าความยาวของข้อมูลที่ตามมามีหน่วยเป็นไบต์
Data	Length bytes	เก็บข้อมูล
repeated any number of times. read first byte to check for terminator		
Terminator	1 byte	ตัวเลขแสดงค่าจบ มีค่า = 0!
ExtensionBlocks	cannot be pre calculated	optional. read first byte to check its presence
format is identical to the ExtensionBlock above		
Terminator	1 byte	ตัวเลขแสดงค่าจบ มีค่า = 3B

ตารางที่ 2.5 อธิบาย Extension Block function codes

Function Code	ชื่อ	ขนาด	รายละเอียด
1	PlaintextExtension	any	< incomplete >
249	LocalDescriptorExtension	4 bytes	
	Flags	1 byte	
	reserved ?	bits 7-5	สงวนไว้มีค่าเป็น 0
	Undraw	bits 4-2	เก็บค่ารหัสต่างๆของภาพ
	User Input	bit 1	
	Transparent	bit 0	=1 รูปภาพมีสีโปรงใส =0 รูปภาพไม่มีสีโปรงใส
	Duration	2 bytes	
	TransparentColor	1 byte	ระบุตำแหน่งของสีที่โปรงใส
254	CommentExtension	any	
255	ApplicationExtension	any	

2.5 การบีบอัดข้อมูลแบบ LZW

การบีบอัดข้อมูลแบบ Lempel Ziv (LZ) ถูกตีพิมพ์ครั้งแรกในปี 1977 และในเวลาต่อมา Terry Welch ได้ทำการพัฒนาวิธีการบีบอัดข้อมูลแบบ LZ ไปเป็นวิธีการแบบ LZW ซึ่งเป็นการบีบอัดข้อมูลที่มีประสิทธิภาพมาก และตีพิมพ์เผยแพร่ในปี 1984

2.5.1 การบีบอัดข้อมูลของ LZW

วิธีคิดการบีบขนาด LZW ในรูปแบบง่ายดายที่สุดจะถูกใช้ในขั้นตอนดังต่อไปนี้ คือ LZW พยายามที่จะ output code ของ string ที่เคยเจออยู่ก่อนหน้าแล้ว ออกไปและเมื่อทำการส่ง code ใหม่ ออกไป ก็จะไปเพิ่มใส่ในส่วนของ string ด้วย

ALGORITHM ของ LZW_COMPRESS

```
STRING = get input character
WHILE there are still input characters DO
    CHARACTER = get input character
    IF STRING+CHARACTER is in the string table then
        STRING = STRING+character
```

```
        ELSE
            output the code for STRING
            add STRING+CHARACTER to the string table
            STRING = CHARACTER
        END of IF
    END of WHILE
    output the code for STRING
```

ตัวอย่าง 2.5.1.1 การบีบอัดข้อมูลด้วยวิธีการ LZW

สมมติมีข้อความ /WED/WE/WEE/WEB/WET ดังต่อไปนี้ จำทำการบีบข้อความนี้ด้วยวิธีการบีบข้อมูลแบบ LZW ดัง อัลกอริทึ่มของการบีบข้อมูลแบบ LZW จะได้ผลดังตารางข้างล่างนี้

ตารางที่ 2.6 แสดงค่าของตัวแปรต่าง ๆ ที่ใช้ในการบีบอัดข้อมูล

Input String = /WED/WE/WEE/WEB/WET			
Character Input	**Code Output**	**New code value**	**New String**
/W	/	256	/W
E	W	257	WE
D	E	258	ED
/	D	259	D/
WE	256	260	/WE
/	E	261	E/
WEE	260	262	/WEE
/W	261	263	E/W
EB	257	264	WEB
/	B	265	B/
WET	260	266	/WET
EOF	T		

จากตารางเราจะได้ ข้อมูลที่ผ่านการบีบอัดแล้วคือ / W E D 256 E 260 261 257 B 260 T

2.5.2 การคลายข้อมูลของ LZW

ขั้นตอนที่ควบคู่กันของการบีบข้อมูล ก็คือขั้นตอนการถอดรหัสข้อมูล โดยการถอดรหัสข้อมูลนั้น จะรับข้อมูลที่ได้จากการบีบมา และจำพยายามทำให้ได้ข้อความต้นฉบับออกมา เหตุผลหนึ่งที่ LZW เป็นที่นิยมใช้และมีประสิทธิภาพ ก็เพราะว่า ไม่ต้องส่งตารางที่ใช้ในการเข้ารหัสไปให้ในกระบวนการถอดรหัส โดยตารางจะถูกสร้างระหว่างขั้นตอนการบีบอัดและถูกแทรกลงไปในส่วนข้อมูลด้วย มีขั้นตอนดังนี้

ALGORITHM ของ LZW_DECOMPRESS

```
Read OLD_CODE
output OLD_CODE
WHILE there are still input characters DO
        Read NEW_CODE
        STRING = get translation of NEW_CODE
        output STRING
        CHARACTER = first character in STRING
        add OLD_CODE + CHARACTER to the translation table
        OLD_CODE = NEW_CODE
END of WHILE
```

ตัวอย่าง 2.5.2.1 การคลายข้อมูลด้วยวิธีการ LZW

เป็นการเอาข้อมูล / W E D 256 E 260 261 257 B 260 T ผ่านการถูกบีบอัด ด้วยวิธี LZW แล้ว มาคลายข้อมูลออก ให้กลายเป็นข้อมูลที่แท้จริง ซึ่งจะได้ผลดังตารางข้างล่างนี้

ตารางที่ 2.7 แสดงค่าของตัวแปรต่าง ๆ ที่ใช้ในการคลายข้อมูล

Input Codes: / W E D 256 E 260 261 257 B 260 T				
Input/ NEW_CODE	**OLD_CODE**	**STRING/ Output**	**CHARACTER**	**New table entry**
/	/	/		
W	/	W	W	256 = /W
E	W	E	E	257 = WE
D	E	D	D	258 = ED
256	D	/W	/	259 = D/
E	256	E	E	260 = /WE
260	E	/WE	/	261 = E/
261	260	E/	E	262 = /WEE
257	261	WE	W	263 = E/W
B	257	B	B	264 = WEB
260	B	/WE	/	265 = B/
T	260	T	T	266 = /WET

จากตารางเราจะได้ข้อความที่คลายออกมาแล้วคือ /WED/WE/WEE/WEB/WET

2.6 ทฤษฎีความน่าจะเป็นในการลดขนาดข้อมูล

ขั้นตอนการลดขนาดของข้อมูล โดยทั่วไปแล้วมีอยู่หลายขั้นตอน โดยจะมีขั้นตอนหลักๆคือ การหาค่าความถี่และความน่าจะเป็นของข้อมูล โดยข้อมูลใดมีการใช้มากจะมีการลดขนาดข้อมูลให้ในแต่ละตัวอักษรมีขนาดของบิตต่ำส่วนข้อมูลที่มีการใช้น้อยจะลดขนาดข้อมูลให้มีขนาดของบิตสูง หากต้องการที่จะรู้ว่าข้อมูลส่วนใดมีการใช้มากหรือใช้น้อย ก็จะใช้ทฤษฎีความน่าจะเป็นทำการตรวจสอบข้อมูล โดยการเก็บค่าความถี่และความน่าจะเป็นของข้อมูลแต่ละตัว ความน่าจะเป็นของเหตุการณ์ใดๆ มีค่าได้ตั้งแต่ 0 ถึง 1 และค่าความน่าจะเป็นคือตัวเลขที่ชี้บอกถึงเหตุการณ์นั้น ๆ ว่ามีโอกาสเกิดขึ้นมากน้อยเพียงใด ถ้ามีค่าใกล้ 0 หมายถึงเหตุการณ์นั้นมีโอกาสเกิดขึ้นน้อยมาก และจะไม่เกิดขึ้นหรือเป็นไปไม่ได้ถ้ามีค่าเป็น 0 ถ้ามีค่าใกล้ 1 หมายถึงเหตุการณ์นั้นมีโอกาสในการเกิดสูง และจะเกิดขึ้นแน่นอนถ้ามีค่าเป็น 1 ค่าความน่าจะเป็นที่ใช้ในการลดขนาดข้อมูลจะสามารถคำนวนได้ดังสมการต่อไปนี้

$$\text{ความน่าจะเป็นของข้อมูลแต่ละตัวอักษร} = \frac{\text{(จำนวนครั้งที่ใช้ข้อมูลแต่ละตัว)}}{\text{(จำนวนครั้งที่ใช้ข้อมูลทั้งหมด)}}$$

การลดขนาดข้อมูลโดยแนวคิดทางคณิตศาสตร์

แนวคิดพื้นฐานในการลดขนาดข้อมูลคือ " ถ้าทราบความน่าจะเป็นของสัญลักษณ์หรือตัวอักษรในข้อมูลชุดหนึ่ง ก็จะสามารถลดขนาดของข้อมูลชุดนั้นได้ "

2.7 ทฤษฎีการเข้ารหัสข้อมูลแบบ Huffman Coding

2.7.1 นิยาม

การเข้ารหัสแบบ Huffman Coding นี้เป็นการเข้ารหัสแบบวิธี Lossless Data Compression ซึ่งเป็นการลดขนาดข้อมูลโดยที่รักษาความถูกต้องของข้อมูลได้ 100% ถูกคิดค้นขึ้นโดย D.A. Huffman และตีพิมพ์ครั้งแรกในบทความ A Method for the Construction of Minimum Redundancy Codes ในปี ค.ศ. 1952 โดยมีขั้นตอนวิธีดังนี้

1. สำหรับข้อมูลชุดหนึ่ง นับจำนวนความถี่หรือความน่าจะเป็นของแต่ละตัวอักษร
2. เรียงลำดับ(sort)ตามความน่าจะเป็นจากน้อยไปมากเก็บไว้ในลิสต์(list)
3. สร้างแผนภูมิต้นไม้แบบ Binary tree ขึ้นมาโดยเริ่มสร้างโหนดขึ้นทีละคู่ จากคู่ของข้อมูลที่มีค่า ความน่าจะเป็นน้อยที่สุดไปหาคู่ของข้อมูลที่มีค่าความน่าจะเป็นมากที่สุด แต่ถ้าข้อมูล

ตัวสุดท้ายไม่สามารถจัดเข้าคู่ได้คือเหลือเพียงตัวเดียว ก็จะถูกแยกเป็นโหนดอิสระที่มีกิ่งเพียงกิ่งเดียว

4. สร้างโหนดแม่(Parent Node) จากโหนดลูก(Child Node)คู่ที่ได้จากขั้นตอนที่ 3 ซึ่งจะมีค่าความน่าจะเป็นเท่ากับผลรวมของค่าความน่าจะเป็นของโหนดลูกทั้งสอง
5. โหนดแม่ที่ได้จะถูกเพิ่มเข้ามาเป็นโหนดอิสระในลิสต์ที่เรียงลำดับจากน้อยไปมาก และโหนดลูกก็จะถูกยกเลิกออกจากลิสต์
6. กำหนดค่าบิตให้กับโหนดลูกคู่ที่รวมเป็นโหนดแม่ในขั้นตอนที่ 4 โดยให้โหนดลูกทางด้านซ้ายมีค่าเป็น 0 ส่วนโหนดลูกทางด้านขวามีค่าเป็น 1
7. ทำซ้ำขั้นตอนที่ 3 ถึง 6 จนกระทั่งเหลือโหนดอิสระเพียงโหนดเดียวซึ่งจะถูกกำหนดให้เป็นโหนดราก (Root Node)

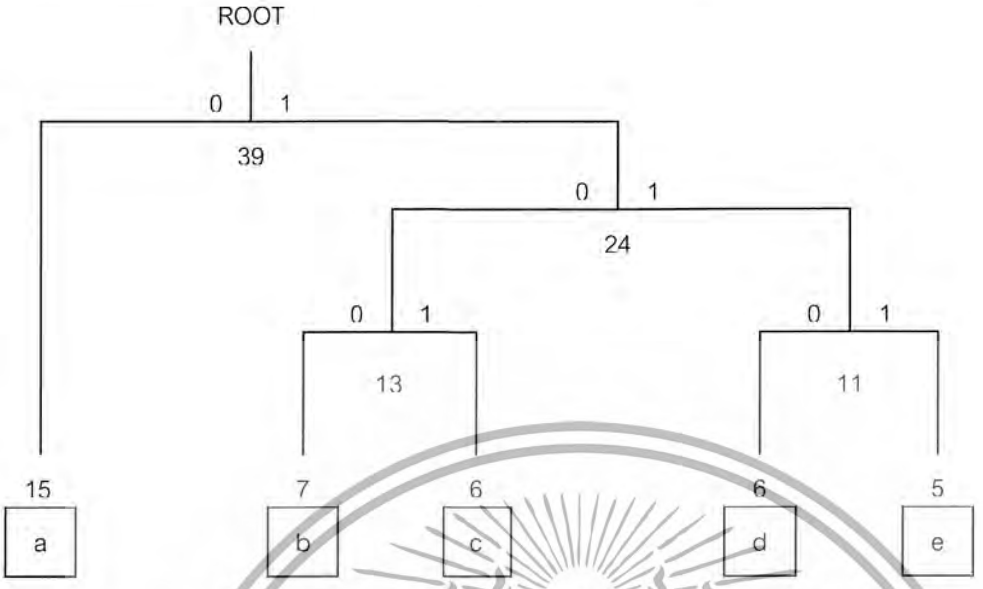
2.7.2 วิธีการเข้ารหัสแบบ Huffman Coding

วิธีการเข้ารหัสแบบนี้จะเริ่มต้นด้วยการนำตัวอักษรทั้งหมดที่ปรากฏอยู่ในข้อความ มาหาค่า Occurrence โดยสมมติว่า มีข้อมูลชุดหนึ่งประกอบด้วยตัวอักษร a ถึง e เป็นจำนวนทั้งหมด 39 ตัวอักษร ซึ่งแต่ละตัวอักษรมีค่าความถี่ต่าง ๆ กัน แสดงค่าความถี่ดังตารางด้านล่าง

ตารางที่ 2.8 แสดงค่าความถี่ของข้อมูล

ตัวอักษร	ความถี่
a	15
b	7
c	6
d	6
e	5

จากตารางค่าความถี่เราจะได้แผนภูมิต้นไม้แบบ Huffman ดังนี้



รูปที่ 2.3 แสดง Huffman Tree ที่ได้ สำหรับข้อมูลเบื้องต้น

ถึงตรงนี้เราก็จะ ได้ผลลัพธ์ของการเข้ารหัสแบบ Huffman Coding โดยจะดูเส้นทางจากส่วนปลายด้านขวาที่จะเรียกกันว่า Root ไปยังส่วนปลายด้านซ้ายที่เป็นตัวอักษรก็จะ ได้รหัสสำหรับตัวอักษรแต่ละตัว ซึ่งแสดงได้ดังนี้

ตารางที่ 2.9 แสดงการเข้ารหัสแบบ Huffman coding

ตัวอักษร	a	b	c	d	e
โค้ด	0	100	101	110	111

จากนั้น ก็แทนตัวอักษรในข้อมูลด้วยบิตข้อมูลที่ได้ เช่น ตัวอักษร a นั้นตามมาตรฐาน ASCII คือ 97 (ฐาน 10) หรือจะเขียนในรูปแบบเลขฐานสองคือ 01100001 แต่หลังจากที่เข้ารหัสตามผลที่ได้ก็จะมีขนาดเพียง 1 บิต คือ 0 การเข้ารหัสแบบนี้จะเป็นการรับประกันว่าจะไม่มีรหัสของตัวอักษรใดๆซ้ำกัน และจะไม่มีรหัสของตัวอักษรใดเป็น Prefix (เช่น ถ้าตัวอักษร a เข้ารหัสเป็น 0 แล้วจะไม่มีตัวอักษรใดขึ้นต้นด้วย 0) ของตัวอักษรอื่น

2.8 ทฤษฎี Steganography

คือศาสตร์ที่ว่าด้วยการซ่อนข้อมูลหนึ่งไว้ในอีกข้อมูลหนึ่งโดยมีวัตถุประสงค์ที่จะปิดบังไม่ให้ผู้อื่นสงสัยได้ว่ามีการซ่อนข้อมูลเอาไว้ โดยทั่วไปแล้วเราจะทำการเข้ารหัสลับ (encrypt) ข้อมูลที่ต้องการจะซ่อนเสียก่อน ก่อนที่จะนำไปซ่อนในข้อมูลอื่น แต่ไม่จำเป็นว่าจะต้องทำการเข้ารหัสลับด้วยเสมอ เพราะการเข้ารหัสลับนั้นทำไปเพื่อเพิ่มระดับความปลอดภัยของข้อมูลที่เราจะซ่อนเท่านั้น

2.8.1 คำเฉพาะในเรื่อง Steganography

- Cover carrier คือ ข้อมูลหรือสื่อที่จะใช้จัดเก็บข้อมูลที่ต้องการจะซ่อน ซึ่ง Cover carrier อาจจะเป็นได้ทั้ง audio, text, image, video หรือสื่ออื่น ๆก็ได้

- Embedded message คือ ข้อมูลที่เราต้องการจะซ่อน ไว้ใน Cover carrier ซึ่งเป็นข้อมูลที่มี การเป็นแบบ Bit stream

- Stego carrier คือผลลัพธ์ที่ได้จากการนำ Embedded message มาฝังลงใน Cover carrier

- Stegokey คือ ข้อมูลที่เป็นความลับ เช่น password ที่นำมาใช้เพื่อให้ข้อมูลมีความปลอดภัยมากยิ่งขึ้น

2.8.2 ข้อควรพิจารณาในการทำ Steganography

1. Cover carrier ไม่ควรที่จะเปลี่ยนแปลงเสียหาย จนทำให้สามารถรับรู้ได้ว่ามีการซ่อน Embedded message เอาไว้

2. Embedded message ควรจะ encode อยู่ที่ตัวข้อมูลจริง ๆ ของ cover carrier ไม่ควรอยู่ที่ header หรือ wrapper ของ cover carrier เพื่อรักษาความสอดคล้องของ cover carrier ให้คงอยู่

3. ควรระวังเรื่องความผิดเพี้ยนเปลี่ยนแปลงของ embedded message เนื่องจากการที่ cover carrier ถูกปรับเปลี่ยน

2.8.3 Steganography in Images

Image Steganography ถูกนำมาใช้ประโยชน์ เนื่องจากความสามารถในการรับรู้ด้วยสายตาของมนุษย์นั้นมีขีดจำกัด โดยระดับความเข้มของสีเดียวกัน ที่แตกต่างกันเพียงเล็กน้อยนั้น จะไม่มีความแตกต่างกันเลยในการรับรู้ของมนุษย์ ด้วยเหตุนี้เราจึงสามารถนำข้อมูลที่จัดเก็บอยู่ในรูปแบบ bit stream เช่น text หรือแม้แต่ image เองมาซ่อนไว้ใน image ที่จัดเก็บอยู่ในคอมพิวเตอร์ได้

2.8.3.1 Image Encoding Technique

วิธีการส่วนใหญ่ที่ใช้ในการซ่อนข้อมูลลงในภาพมีอยู่ 3 วิธี คือ

1. Least Significant Bit (LSB) insertion
2. Masking and Filtering Technique
3. Algorithms and Transformations

ในที่นี้จะกล่าวถึงเฉพาะวิธีที่นำมาใช้ในโครงงาน

Least Significant Bit (LSB) insertion

วิธีนี้เป็น Image Steganography Technique ที่รู้จักกันมากที่สุดวิธีหนึ่ง โดยมีหลักการทำงาน คือ นำข้อมูลแต่ละ bit ของ embedded message เข้าไปเก็บที่ least significant bit ของแต่ละ byte ข้อมูลของ cover image ข้อดีของหลัก LSB insertion คือ ข้อมูลสามารถซ่อนเข้าไปใน least bit ได้โดยไม่มีผลกระทบต่อการรับรู้ของมนุษย์

โดยถ้าเป็น image แบบ 24-bits-per-pixel แสดงว่าใน 1 pixel เราสามารถที่จะ encode ส่วนของข้อมูลที่เป็น embedded message เข้าไปได้ 3 bit นั่นคือถ้าเราต้องการ embed ตัวอักษร A ซึ่งมี binary value เป็น 01000001 เข้าไปเก็บใน 24-bit image เราต้องใช้ทั้งหมด 3 pixel ดังแสดงข้างล่าง

ข้อมูลของภาพก่อนนำตัวอักษร A เข้าไปซ่อน

(00100111 11101001 11001000) (00100111 11001000 11101001)

(11001000 00100111 11101001)

นำค่า binary value ของตัวอักษร A (01000001) ซ่อนเข้าไปใน 3 pixel ข้างต้น โดยเริ่มจาก top left byte จะได้ผลดังนี้

(00100110 11101001 11001000) (00100110 11001000 11101000)

(11001000 00100111 11101001)

จะเห็นได้ว่ามีเพียงข้อมูลใน bit ที่เป็นตัวเอียงเท่านั้นที่มีการเปลี่ยนแปลง

ถ้า cover image เป็นภาพแบบการใช้ตารางสี การใช้วิธี LSB insertion กับข้อมูลภาพนั้นควรระลึกไว้เสมอว่าการเปลี่ยนแปลงค่าข้อมูลนั้นแม้เพียง 1bit อาจหมายถึงการเปลี่ยนจากเฉดสีหนึ่งไปเป็นอีกเฉดสีหนึ่งเลยทีเดียว ซึ่งการเปลี่ยนแปลงในลักษณะนี้ สายตามนุษย์สามารถรับรู้ความผิดปกติได้อย่างแน่นอน ด้วยเหตุนี้เองวิธี LSB insertion จึงมักจะไม่นำมาใช้กับภาพที่มีลักษณะเป็นภาพ Grayscale

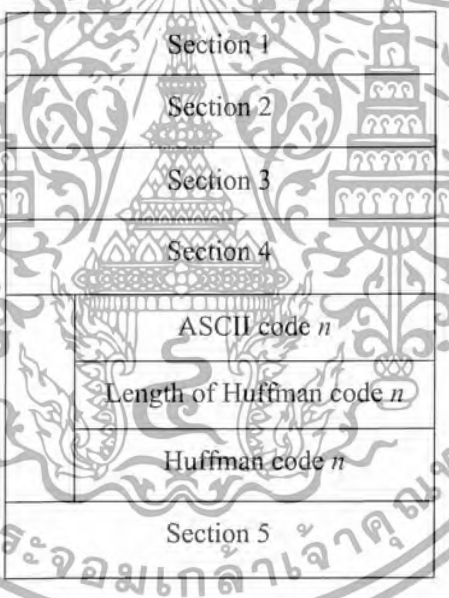
บทที่ 3
ขั้นตอนการดำเนินการ

3.1 Header ของข้อมูลที่นำไปซ่อน

ข้อความที่จะนำไปซ่อนลงในรูปภาพ จะถูกผ่านการบีบอัดข้อมูลแบบ Huffman Coding เพื่อลดขนาดของข้อความให้มีขนาดเล็กลง โดยข้อความที่ผ่านการเข้ารหัสแล้ว ยังไม่สามารถนำไปซ่อนได้โดยตรง เนื่องจากไม่สามารถที่จะถอดรหัสกลับมาเป็นข้อความเดิมได้ จึงต้องมีส่วนของ Header ซึ่งประกอบด้วยส่วนต่างๆที่เป็นลักษณะเฉพาะของข้อความที่ผ่านการเข้ารหัสแล้ว

3.1.1 Header

Header ของข้อความที่จะนำไปซ่อนนั้นจะประกอบด้วยส่วนต่างๆหลายส่วน จะอธิบายได้ดังรูปภาพต่อไปนี้



รูปที่ 3.1 โครงสร้างของ Header of Huffman Coding

โดยแต่ละส่วนของ Header จะอธิบายได้ดังนี้

Section 1

มีขนาด 2 ไบต์ เป็นส่วนที่ใช้ระบุว่าไฟล์ภาพนั้นได้มีการซ่อนข้อมูลไว้ หรือไม่ซึ่งโดยปกติจะมีค่าเท่ากับYP

Section 2

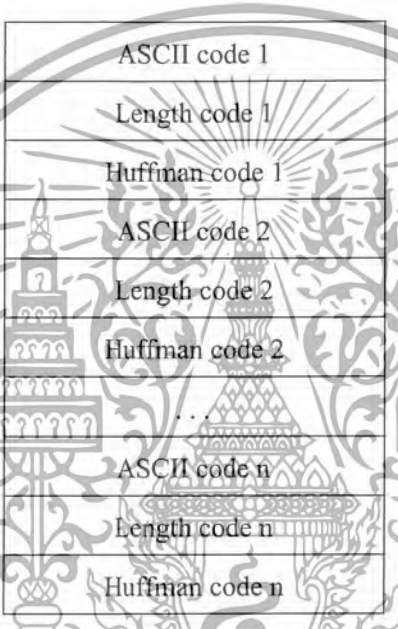
มีขนาด 3 ไบต์ จะเก็บส่วนความยาวของข้อมูลที่ถูกเข้ารหัสด้วย Huffman Coding ที่จะเอาไปซ่อนลงในไฟล์รูปภาพว่ามีขนาดความยาวกี่ตัว

Section 3

มีขนาด 1 ไบต์ ส่วนนี้จะเก็บความแตกต่างของตัวอักษรที่ปรากฏในข้อความต้นฉบับว่าใช้ตัวอักษรที่แตกต่างกันทั้งหมดกี่ตัว และในนี้จะมี

Section 4

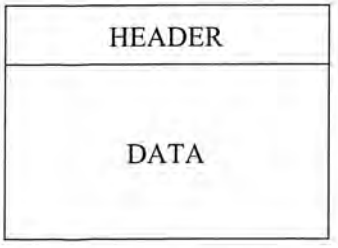
มีขนาดไม่แน่นอน จะเก็บตาราง Huffman Coding ลักษณะการเก็บข้อมูลจะมีรูปแบบดังนี้



รูปที่ 3.2 แสดง การเก็บข้อมูลใน Header of Huffman coding ส่วนที่ 4

Section 5

มีขนาดไม่แน่นอน ซึ่งในส่วนนี้จะเก็บเนื้อข้อความที่ถูกแปลงโดยการเข้ารหัสแบบ Huffman Coding ดังนั้นข้อมูลที่จะนำไปซ่อนจะประกอบด้วย 2 ส่วนที่เป็น Header และส่วนที่เป็น Data ซึ่งในการซ่อนจะมีขั้นตอนแตกต่างกันในภาพแต่ละชนิด

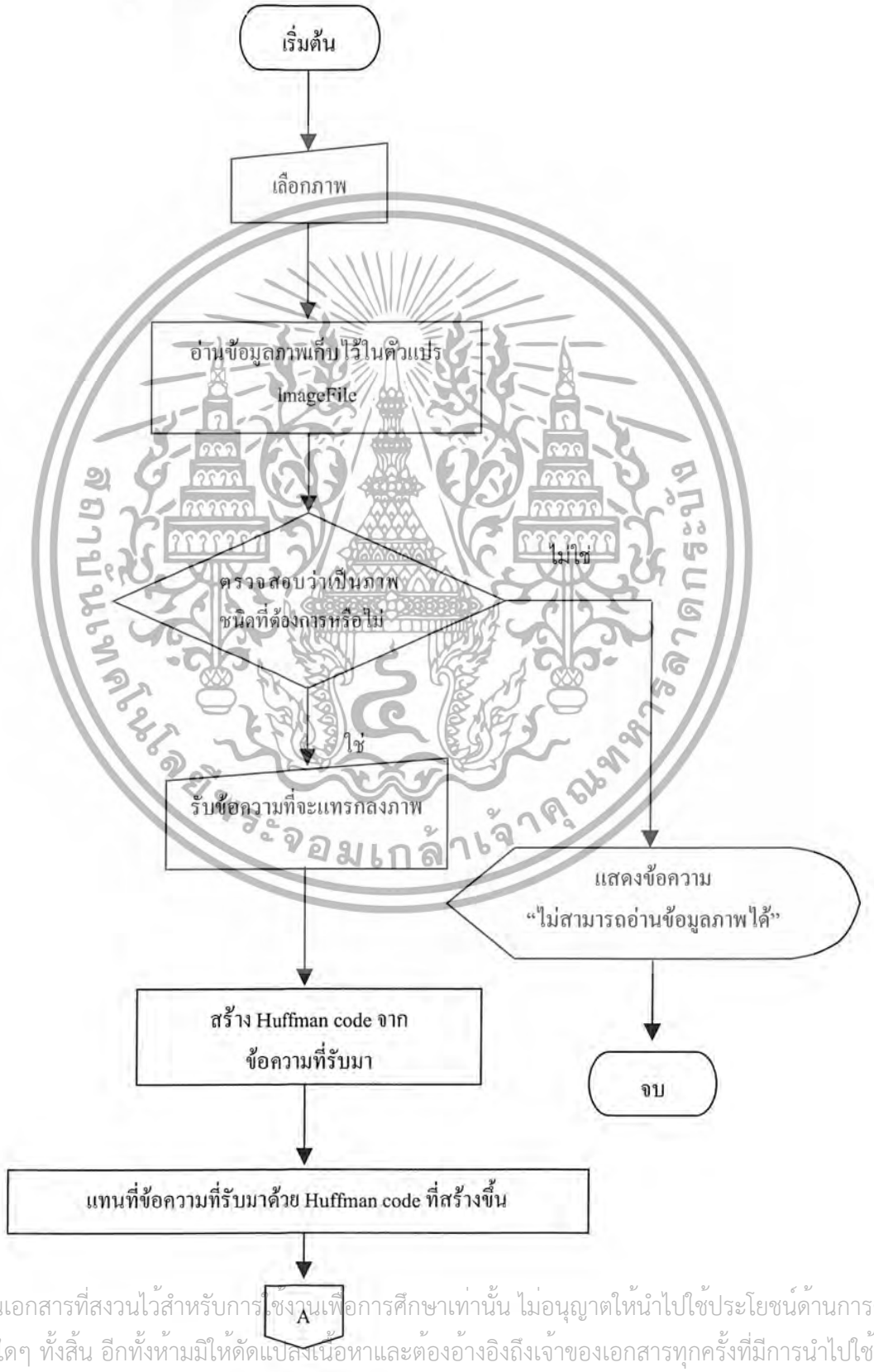


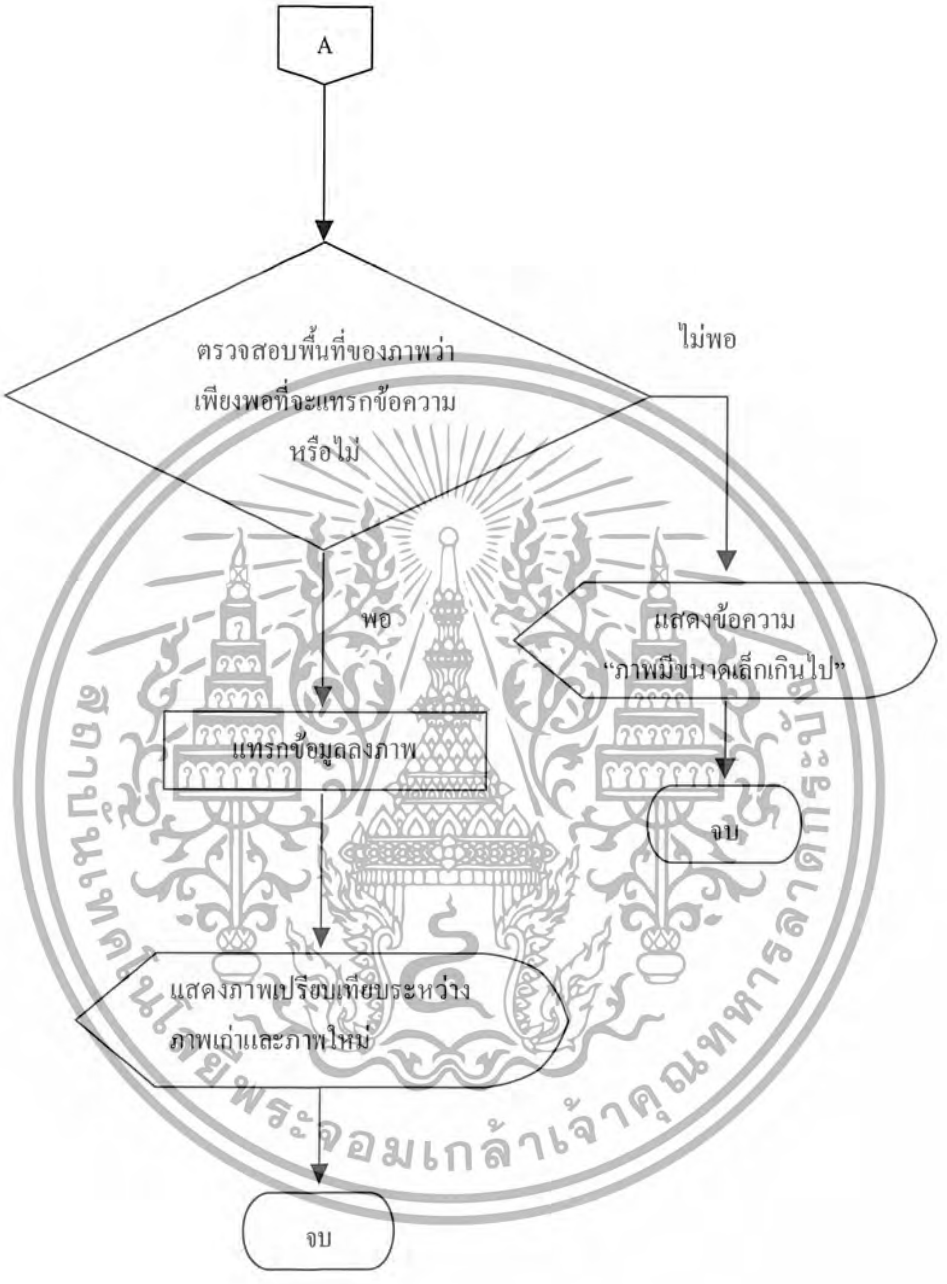
รูปที่ 3.3 รูป Header ของ Huffman

3.2 การซ่อนข้อมูลลงในภาพ

การซ่อนข้อมูลลงในรูปภาพในโครงงานปัญหาพิเศษนี้ มีขั้นตอนการซ่อนข้อมูล ซึ่งจะใช้การอธิบายเป็นแผนผังรูปภาพ ดังต่อไปนี้

Flow การแทรกข้อมูลลงภาพ

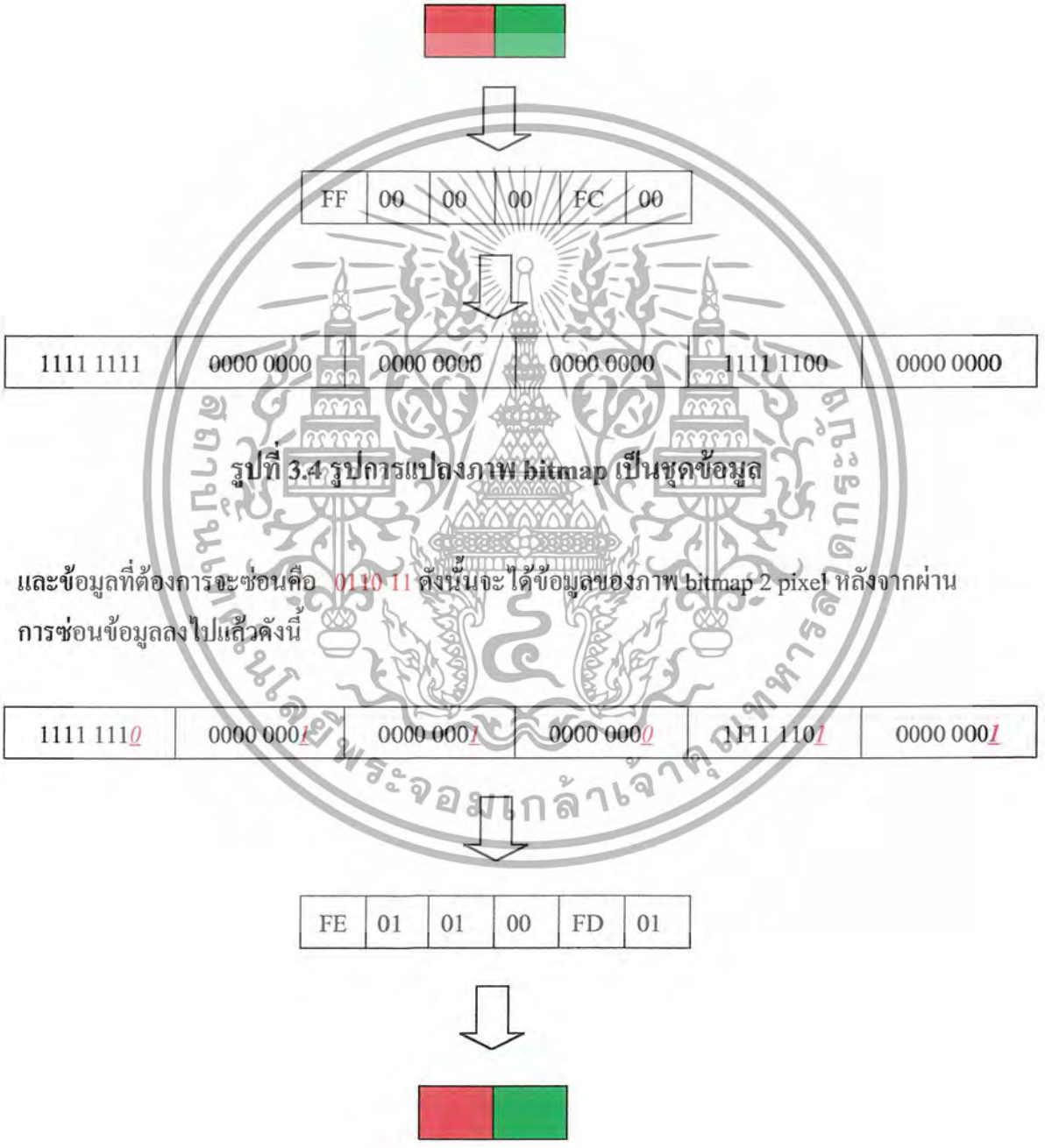




3.2.1 การซ่อนข้อมูลลงในภาพ 24-bit bitmap

เนื่องจากข้อมูลภาพแบบ 24-bit bitmap นั้นใน 1 pixel จะประกอบด้วยข้อมูลที่จะแสดงแบบ RGB อย่างละ 8 บิต ดังนั้นจะได้ว่าใน 1 pixel เราสามารถซ่อนข้อมูลได้ 3 บิต

ตัวอย่าง 3.2.1.1 ถ้าเรามีข้อมูลแบบ bitmap 2 pixel ดังรูป



รูปที่ 3.4 รูปการแปลงภาพ bitmap เป็นชุดข้อมูล

และข้อมูลที่ต้องการจะซ่อนคือ 0110 11 ดังนั้นจะได้ข้อมูลของภาพ bitmap 2 pixel หลังจากผ่านการซ่อนข้อมูลลงไปแล้วดังนี้

รูปที่ 3.5 รูปการซ่อนข้อมูลลงในภาพ Bitmap

จะได้ว่า หลังจากผ่านการซ่อนข้อมูลแล้ว สีของภาพ bitmap จะเปลี่ยนแปลงไปน้อยมาก จนสายตาของมนุษย์ไม่สามารถแยกออกได้

3.2.2 การซ่อนข้อมูลลงในภาพ GIF

เนื่องจากรูปภาพแบบ GIF ในส่วนที่เป็นข้อมูลของภาพไม่ได้เก็บค่าของสีจริง แต่เก็บเลขดรรชนีของสีเท่านั้น ดังนั้นการเปลี่ยนแปลงแก้ไขข้อมูลของภาพอาจจะทำให้สีของภาพเปลี่ยนไปอย่างมากซึ่งแตกต่างจากภาพ bitmap ที่สามารถเปลี่ยนแปลงค่าสีได้โดยตรง

ตัวอย่าง 3.4.2.1 การซ่อนข้อมูลภาพแบบ GIF

สมมติให้จานสีของ GIF ดังกล่าวมีค่าดังต่อไปนี้

Index 0 เก็บค่า 03 01 00 = ดำ 1

Index 1 เก็บค่า FF FF FF = ขาว

Index 2 เก็บค่า FD 04 06 = แดง

Index 3 เก็บค่า 00 00 00 = ดำ 2

และข้อมูลภาพในส่วนของ Raster Data Block หลังจากผ่านขั้นตอนการถอดรหัส LZW แล้ว คือ 01 02 00 03 02 00 01 03

ขาว	แดง	ดำ 1	ดำ 2
แดง	ดำ 1	ขาว	ดำ 2

รูปที่ 3.6 แสดงข้อมูลแบบ GIF

ข้อมูลที่ต้องการซ่อนคือ 0110 ดังนั้นข้อมูลของภาพหลังจากผ่านการซ่อนข้อมูล 0110 ด้วยขั้นตอนที่ใช้กับภาพแบบ 24-bit bitmap จะได้ข้อมูลภาพใหม่ดังนี้ 00 01 01 02 02 00 01 03 โดยได้จาก

ข้อมูลเก่า

01 → 0000 0001 + 0 → 0000 0000 → 00

02 → 0000 0010 + 1 → 0000 0001 → 01

00 → 0000 0000 + 1 → 0000 0001 → 01

03 → 0000 0011 + 0 → 0000 0010 → 02

ข้อมูลใหม่

ดำ 1	ขาว	ขาว	แดง
แดง	ดำ 1	ขาว	ดำ 2

รูปที่ 3.7 แสดงการซ่อนข้อมูลแบบ GIF หลังจากซ่อนข้อมูลลงไป

จะเห็นได้ว่าภาพที่ได้หลังจากผ่านการซ่อนข้อมูลแล้ว แตกต่างไปจากเดิมอย่างเห็นได้ชัด ดังนั้นเราจึงไม่สามารถใช้วิธีการซ่อนแบบ bitmap ได้ ดังนั้นในการซ่อนข้อความลงไปในภาพ GIF จะต้องระวังไม่ให้ค่าสีเปลี่ยนแปลงไปอย่างเห็นได้ชัด โดยที่หลักการคือเมื่อต้องการซ่อนข้อความลงไปในภาพเราจะต้องหาค่า index สีที่ใกล้เคียงกันกับสีที่เราจะซ่อน โดยจากภาพตัวอย่างจะได้ว่า

สี index 0 มีค่าเป็น ดำ 1 สามารถจับคู่ได้กับสี index 3 ซึ่งเป็นสี ดำ 2

สี index 1 ไม่สามารถจับคู่ได้

สี index 2 ไม่สามารถจับคู่ได้

สี index 3 มีค่าเป็น ดำ 2 สามารถจับคู่ได้กับสี index 0 ซึ่งเป็นสี ดำ 1

<u>ตัวอย่าง 3.4.2.2</u> การซ่อนข้อมูล 0110

ข้อมูลเก่า

01	0000 0001	0000 0001	01
02	0000 0010	0000 0010	02
00	0000 0000 + 0	0000 0000	00
03	0000 0011 + 1	0000 0000	03
02	0000 0010	0000 0010	02
00	0000 0000 + 1	0000 0011	03
01	0000 0001	0000 0001	01
03	0000 0011 + 0	0000 0000	00

ข้อมูลใหม่

ดังนั้นจะได้ข้อมูลใหม่คือ 01 02 00 03 02 03 01 00

ขาว	แดง	ดำ 1	ดำ 2
แดง	ดำ 2	ขาว	ดำ 1

รูปที่ 3.8 แสดงการซ่อนข้อมูลแบบ GIF หลังมีการซ่อนข้อมูลลงไป

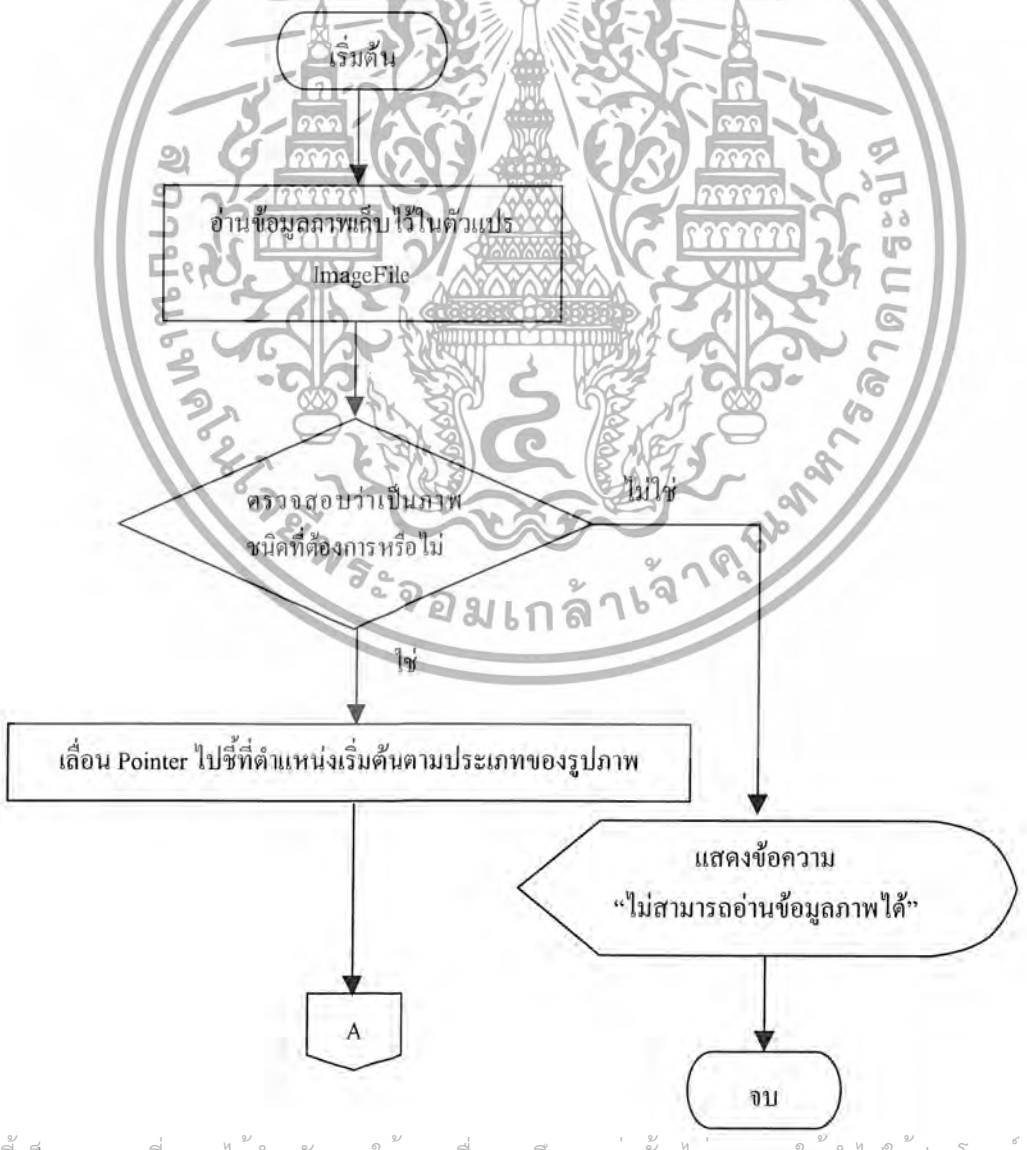
ซึ่งจะเห็นได้ว่า ภาพหลังจากซ่อนข้อมูลแล้วมีการเปลี่ยนแปลงไปจากเดิม แต่ตาของมนุษย์ไม่สามารถที่จะแยกแยะได้ โดยอ้างอิงจากทฤษฎี Steganography ถือว่าการเปลี่ยนแปลงที่เกิดขึ้นนั้นสามารถที่จะยอมรับได้

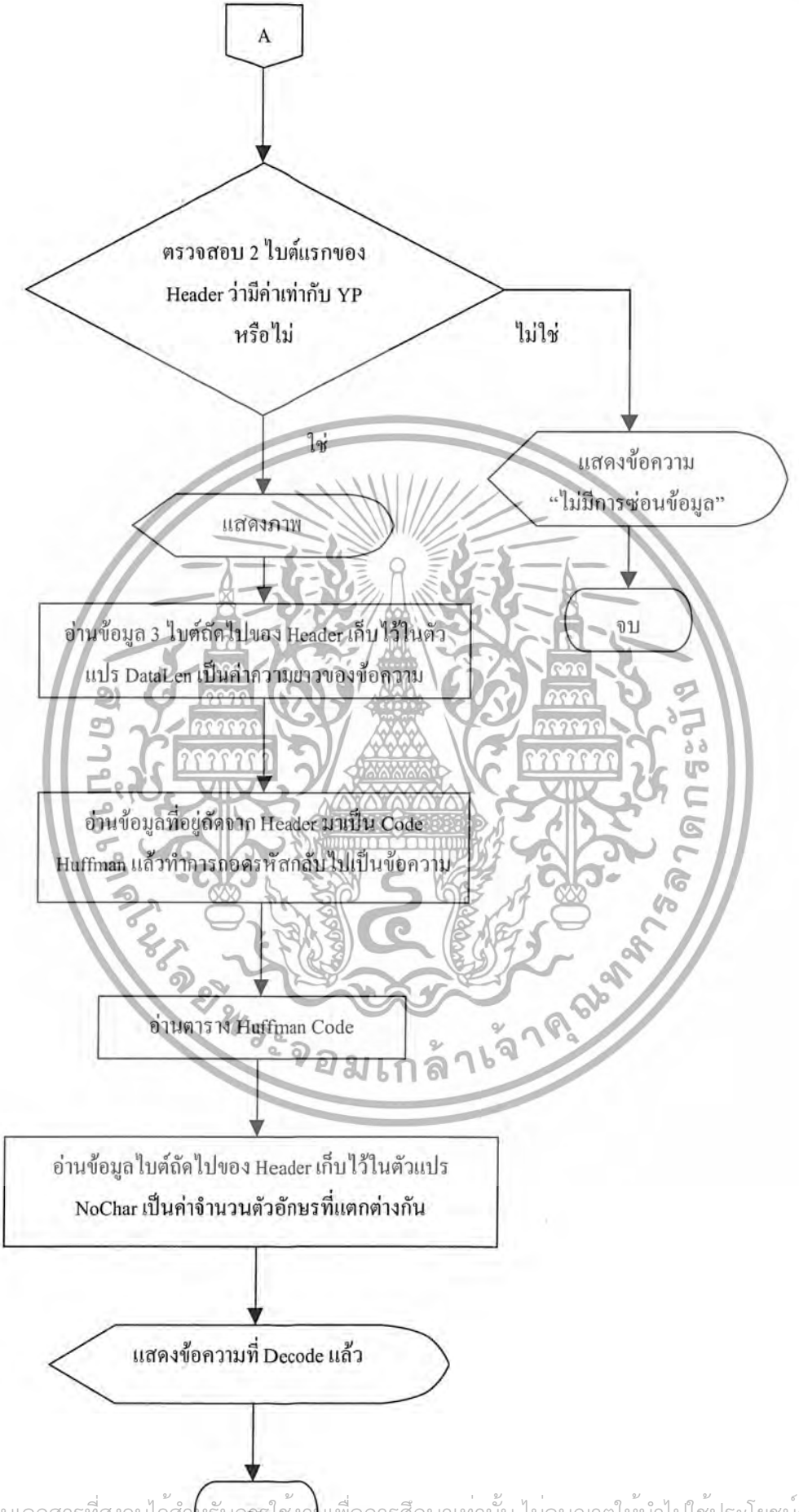
เพราะฉะนั้นภาพ GIF ที่มีการซ่อนข้อมูล 0110 ซึ่งจะได้ชุดข้อมูล 01 02 00 03 02 03 01 00 จากนั้นนำชุดข้อมูลนี้ไปผ่านการเข้ารหัสแบบ LZW แล้วนำไปเก็บไว้ใน Raster Data Block

3.3 การดึงข้อมูลจากภาพ

ขั้นตอนวิธีที่จะดึงข้อมูลที่ถูกซ่อนเอาไว้จากในรูปภาพที่ใช้ในโครงงานปัญหาพิเศษนี้ จะใช้ขั้นตอนดังที่จะแสดงในแผนผังขั้นตอนข้างล่างดังต่อไปนี้

Flow การดึงข้อมูลจากภาพ





3.3.1 การดึงข้อมูลจากภาพ 24-bit bitmap

หลังจากที่มีการซ่อนข้อมูลลงในภาพแบบ 24-bit bitmap เมื่อต้องการเปิดภาพเพื่อดึงข้อความออกมา เราจะเริ่มดึงข้อมูลตั้งแต่ไบต์ที่ 54 เป็นต้นไป (เหตุผลได้กล่าวไปแล้วในขั้นตอนการซ่อนข้อมูลลงภาพ 24 bit bitmap)

ตัวอย่าง 3.5.1.1 ภาพแบบ 24-bit bitmap ที่มีข้อมูลตั้งแต่ไบต์ที่ 54 เป็นต้นไปคือ 62 01 00 FE 00 24 EE 11 8A 5D 20 5B E0 3A 51 6E 70 10 . . . นั้นคือ มีค่าเท่ากับเลขฐานสอง

ตารางที่ 3.1 ตารางแสดงค่าเลขฐานสองของชุดข้อมูล

ฐาน 16	ฐาน 2
62	0110 0010
01	0000 0001
00	0000 0000
FE	1111 1110
00	0000 0000
24	0010 0100
EE	1110 1110
11	0001 0001
8A	1000 1010
5D	0101 1101
20	0010 0000
5B	0101 1011
E0	1110 0000
3A	0011 1010
51	0101 0001
6E	0110 1110
70	0111 0000
10	0001 0000

จากข้อมูลดังกล่าว ในการหาค่าที่ซ่อนไว้เอาไว้เราจะดึงบิตที่ต่ำที่สุด (LSB) ของทุกๆ ไบต์ข้อมูลนั้นขึ้นมา (เนื่องจากเราใช้การซ่อนข้อมูลแบบ LSB) ดังนั้นค่าที่ดึงขึ้นมาได้จะมีค่าเท่ากับ 0 1 0 0 0 0 0 1 0 1 0 1 0 0 1 0 0 0 .... โดยค่า 0 1 0 0 0 0 0 1 จะมีค่าเท่ากับตัวอักษร “A” และค่า 0 1 0 1 0 0 1 0 จะมีค่าเท่ากับตัวอักษร “R” ซึ่งค่าที่ได้มีค่าไม่เท่ากับ “YP” ดังนั้นข้อมูลภาพ Bitmap ที่ยกตัวอย่าง จะบอกได้ว่าภาพนั้นไม่มีการซ่อนข้อมูลอยู่

3.3.2 การดึงข้อมูลจากภาพ GIF

จากตัวอย่างที่ 3.4.2.1 ของภาพ GIF จะได้ว่า

สี index 0 มีค่าเป็น ดำ 1 สามารถจับคู่ได้กับสี index 3 ซึ่งเป็นสี ดำ 2

สี index 1 ไม่สามารถจับคู่ได้

สี index 2 ไม่สามารถจับคู่ได้

สี index 3 มีค่าเป็น ดำ 2 สามารถจับคู่ได้กับสี index 0 ซึ่งเป็นสี ดำ 1

สมมติให้ ข้อมูลภาพที่ผ่านการถอดรหัสแบบ LZW มาแล้ว คือ 01 02 00 03 02 03 01 00 02 01 นั้นคือ

01 → index 1 ไม่สามารถจับคู่ได้

02 → index 2 ไม่สามารถจับคู่ได้

00 → index 0 สามารถจับคู่กับ index 3 ได้ สามารถดึงข้อมูลออกมาได้เป็น 0

03 → index 3 สามารถจับคู่กับ index 0 ได้ สามารถดึงข้อมูลออกมาได้เป็น 1

02 → index 2 ไม่สามารถจับคู่ได้

03 → index 3 สามารถจับคู่กับ index 0 ได้ สามารถดึงข้อมูลออกมาได้เป็น 1

01 → index 1 ไม่สามารถจับคู่ได้

00 → index 0 สามารถจับคู่กับ index 3 ได้ สามารถดึงข้อมูลออกมาได้เป็น 0

02 → index 2 ไม่สามารถจับคู่ได้

01 → index 1 ไม่สามารถจับคู่ได้

เพราะฉะนั้นจะได้ว่า ข้อมูลที่ถูกซ่อนอยู่คือ 0110 ซึ่งสามารถถอดออกมาได้ตรงกับข้อมูลที่ได้ทดลองทำการซ่อนลงไปดังตัวอย่างที่ 3.4.2.1

3.4 การถอดรหัสข้อมูลจากภาพ

ข้อมูลที่ซ่อนอยู่ในรูปภาพนั้นจะประกอบด้วยส่วนที่เป็น Header และข้อความที่ผ่านการเข้ารหัสด้วย Huffman coding ดังนั้นในการที่จะดึงข้อความออกจากรูปภาพจะต้องทำการถอดรหัสด้วย

ตัวอย่าง 3.4.1 การถอดรหัสข้อมูลที่ซ่อนไว้ในภาพ

สมมติให้ข้อมูลที่ดึงออกมาจากภาพนั้น มีค่าเป็น 01011001 01010000 00000000 00000000 00000110 0000011 01010100 00000001 0 01010111 00000010 10 01000001 00000010 11 010110

ฉะนั้นเมื่อเทียบกับโครงสร้างของ Header ของภาพ เราจะได้ว่า

ส่วนที่ 1 มีขนาด 2ไบต์ มีค่าเท่ากับ 01011001 01010000 โดย 01011001 จะมีค่าเป็นตัวอักษร "Y" และค่า 01010000 จะมีค่าเป็นตัวอักษร "P" ซึ่งค่า "YP" ที่ได้นี้จะตรงกับค่ามาตราฐานของHeader

ส่วนที่ 2 มีขนาด 3 ไบต์ มีค่าเท่ากับ 00000000 00000000 00000110 นั้นคือ ค่าความยาวของข้อความที่ผ่านการเข้ารหัสด้วย Huffman Coding มีค่าเท่ากับ 6

ส่วนที่ 3 มีขนาด 1 ไบต์ มีค่าเท่ากับ 0000011 นั้นคือ มีจำนวนตัวอักษรที่แตกต่างกัน 3 ตัว

ส่วนที่ 4 มีขนาดไม่แน่นอน เป็นส่วนของตาราง Huffman ซึ่งมีค่าดังตาราง

ตารางที่ 3.2 แสดงส่วนของตาราง Huffman

ASCII code	Length of Huffman coding	Huffman code
01010100 ซึ่งมีค่าเท่ากับตัวอักษร T	00000001 = 1	0
01010111 ซึ่งมีค่าเท่ากับตัวอักษร W	00000010 = 2	10
01000001 ซึ่งมีค่าเท่ากับตัวอักษร A	00000011 = 2	11

ส่วนที่ 5 มีขนาดความยาวเท่ากับส่วนที่สอง โดยในที่นี้มีค่าความยาวเท่ากับ 6

ซึ่งสามารถถอดรหัสออกมาได้เป็นค่าดังแสดงในตาราง

ตารางที่ 3.3 แสดงการถอดรหัสของ Huffman coding

Huffman Code	0	10	11	0
ค่าตัวอักษร	T	W	A	T

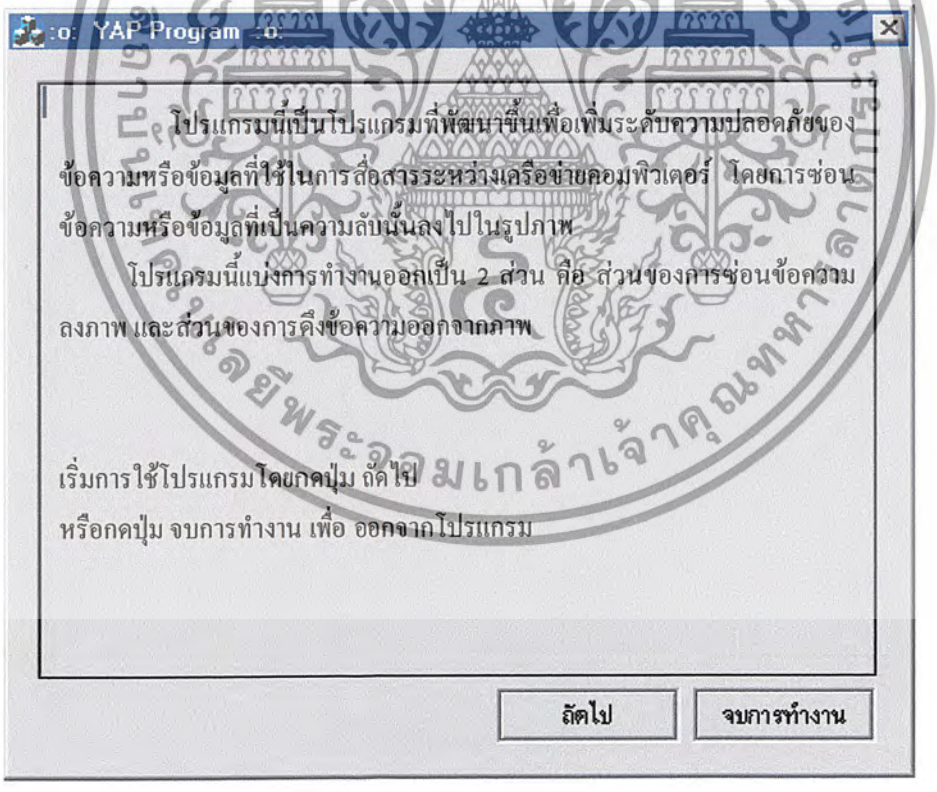
เพราะฉะนั้น เราจะได้ว่าข้อความที่ต้องการคือ TWAT

3.5 ลักษณะการใช้โปรแกรม

เมื่อเปิดโปรแกรมนี้ขึ้นมา จะพบหน้าต่างแรกซึ่งเป็นคำแนะนำอธิบายลักษณะ และการทำงานของโปรแกรมและที่ด้านล่างของหน้าต่างจะมีปุ่มกด 2 ปุ่ม คือ ปุ่มถัดไป และปุ่มจบการทำงานโดย

ปุ่มถัดไป คือ ให้โปรแกรมแสดงส่วนการทำงานถัดไป

ปุ่มจบการทำงาน คือ การออกจากโปรแกรม



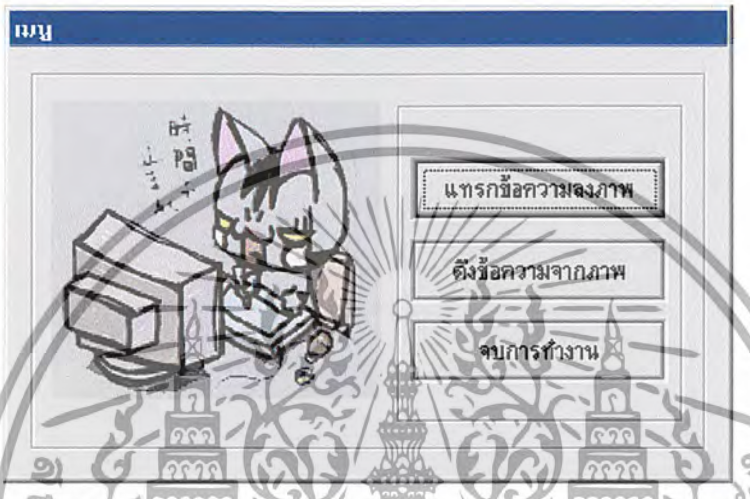
รูปที่ 3.9 แสดงหน้าต่างแนะนำโปรแกรม

หน้าต่างที่ 2 ของโปรแกรมจะเป็นเมนูหลักของโปรแกรม โดยจะมีปุ่มให้ผู้ใช้กด 3 ปุ่มคือ ปุ่มซ่อนข้อความลงภาพ ,ปุ่มดึงข้อความจากภาพ และปุ่มจบการทำงาน โดยแต่ละปุ่มจะมีการทำงาน คือ

ปุ่มซ่อนข้อความลงภาพแสดงส่วนของการซ่อนข้อความลงภาพ

ปุ่มดึงข้อความจากภาพแสดงส่วนของการดึงข้อความออกจากภาพ

ปุ่มจบการทำงานการออกจากโปรแกรม



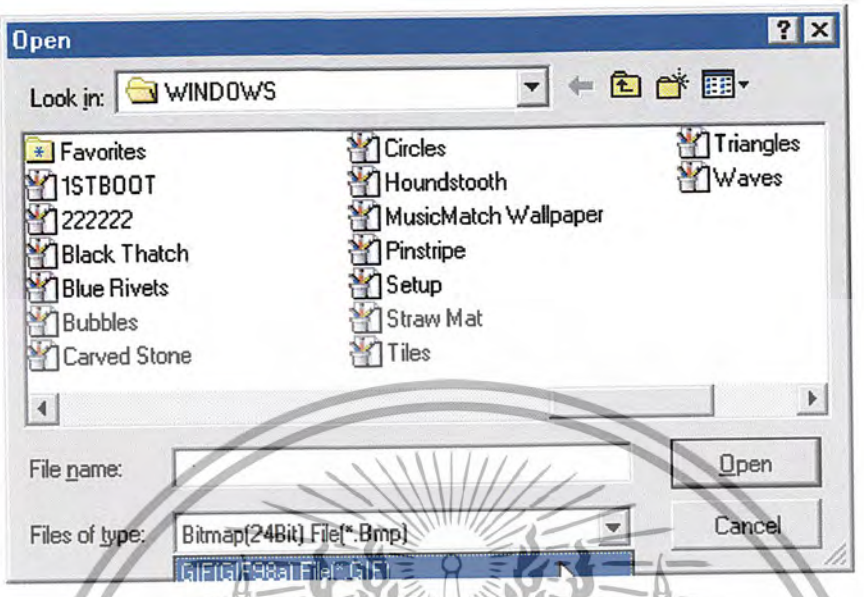
รูปที่ 3.10 แสดงเมนูหลักของโปรแกรม

3.5.1 ส่วนของการซ่อนข้อความลงภาพ

หน้าต่างการทำงานในส่วนนี้แบ่งออกเป็น 2 ส่วนใหญ่ ๆ คือ ส่วนของ รูปภาพ และส่วนของข้อความที่ต้องการซ่อน

ส่วนของรูปภาพจะมีปุ่มเปิดภาพจากแฟ้ม คือปุ่มเลือกรูปภาพที่จะนำมาใช้ในการทำงาน เมื่อกดปุ่มนี้โปรแกรมจะแสดงหน้าต่างให้ผู้ใช้เลือกรูปภาพ โดยที่สามารถเลือกรูปแบบของภาพได้ 2 รูปแบบ คือ รูปภาพแบบ BMP 24 bit และ รูปภาพแบบ GIF

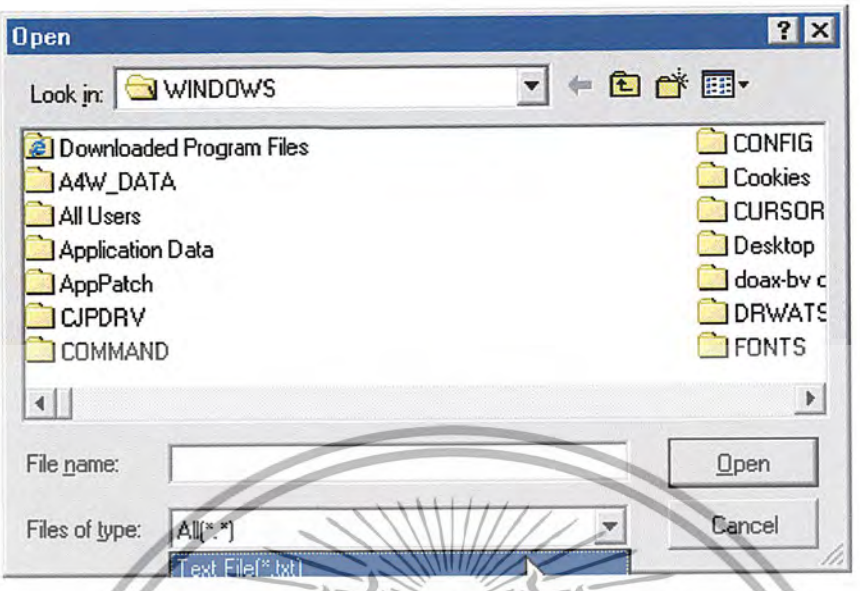
รูปที่ 3.11 แสดงหน้าต่างส่วนของการซ่อนข้อความลงภาพ



รูปที่ 3.12 แสดงหน้าต่างการเปิดรูปภาพ

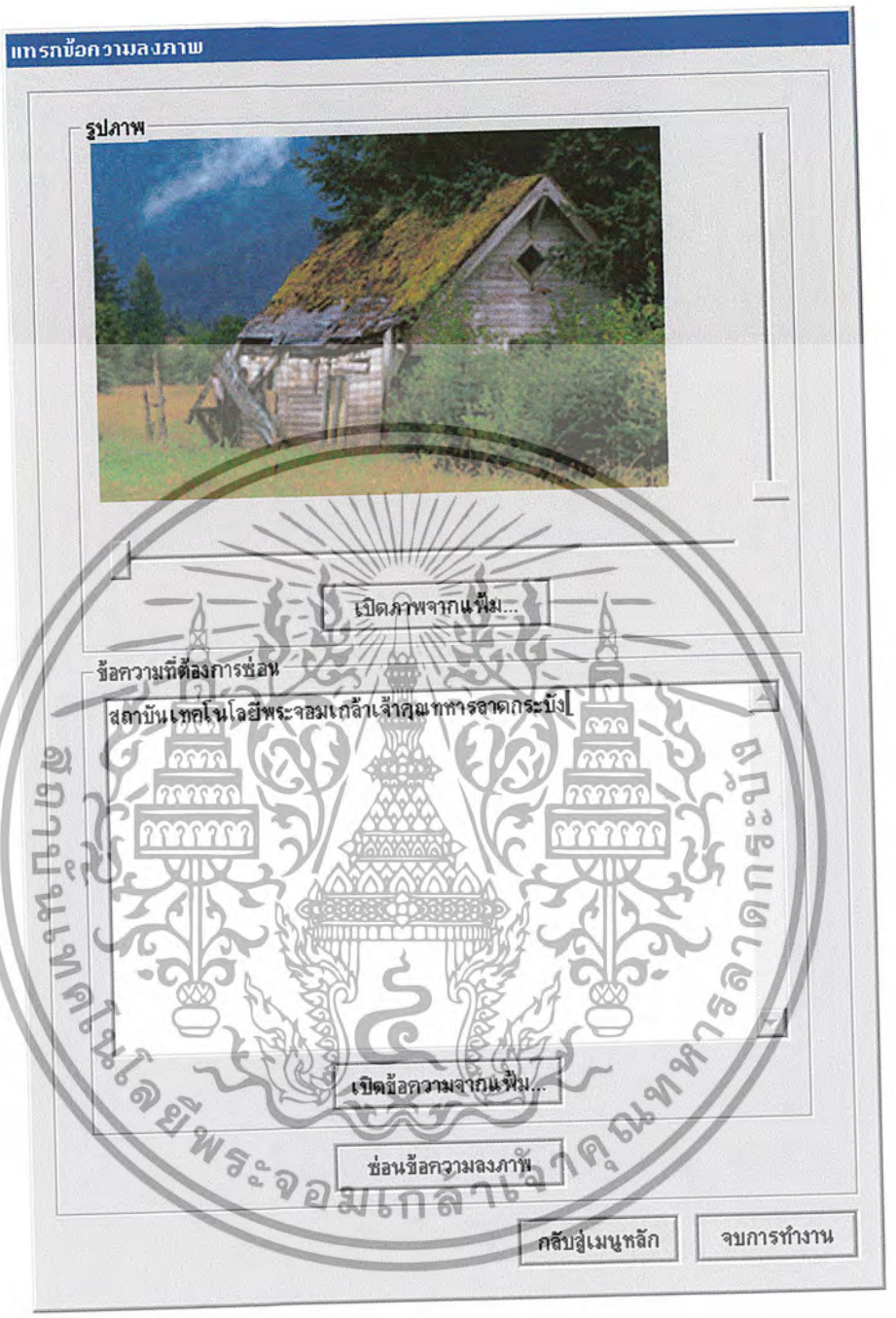
รูปที่ 3.13 แสดงข้อความแจ้งข้อผิดพลาดในกรณีที่ไม่สามารถอ่านข้อมูลจากรูปภาพได้

ส่วนของข้อความ ผู้ใช้สามารถเลือกที่จะพิมพ์ข้อความเองในช่องที่เตรียมไว้ หรือกดปุ่มเปิดข้อความจากแฟ้ม เพื่อเลือกข้อความจากแฟ้มที่มีอยู่แล้วก็ได้



รูปที่ 3.14 แสดงหน้าต่างการเปิดข้อความจากแฟ้ม

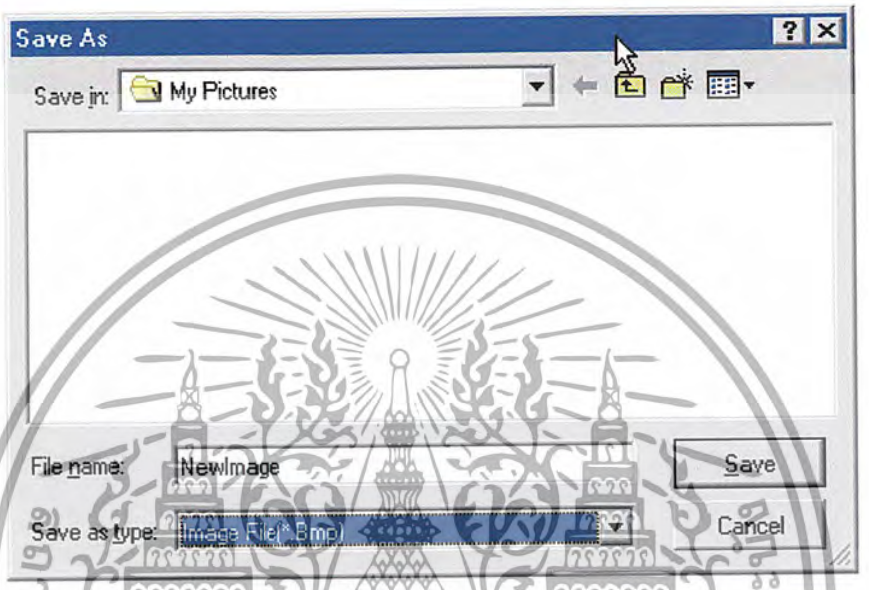
เมื่อได้รูปภาพและข้อความที่จะนำมาซ่อนแล้วกดปุ่มซ่อนข้อความลงภาพ เพื่อให้ โปรแกรมทำการนำข้อความไปซ่อนลงใน ไฟล์ภาพ และเมื่อผู้ใช้กดปุ่ม ซ่อนข้อความลงภาพแล้ว โปรแกรมจะทำการบีบอัดข้อความแล้วซ่อนลง ไปในภาพจากนั้นจะมีหน้าต่างขึ้นมาให้ผู้ใช้บันทึกข้อมูลภาพใหม่



รูป 3.15 แสดงตัวอย่างการพิมพ์ข้อความที่ต้องการซ่อน

โดยกำหนดให้ข้อความที่ต้องการซ่อนคือ “สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง”

เมื่อได้รูปภาพและข้อความที่จะนำมาซ่อนแล้วกดปุ่มซ่อนข้อความลงภาพ เพื่อให้โปรแกรมทำการนำข้อความไปซ่อนลงใน ไฟล์ภาพ และเมื่อผู้ใช้กดปุ่ม ซ่อนข้อความลงภาพ แล้ว โปรแกรมจะทำการบีบอัดข้อความแล้วซ่อนลงไปในภาพจากนั้นจะมีหน้าต่างขึ้นมาให้ผู้ใช้บันทึกข้อมูลภาพใหม่ ในที่นี้ทำการบันทึกข้อมูลภาพใหม่ชื่อ new



รูปที่ 3.16 แสดงหน้าต่างการบันทึกข้อมูลภาพใหม่

หลังจากที่ผู้ใช้บันทึกข้อมูลภาพแล้วโปรแกรมจะแสดงข้อความแจ้งให้ผู้ใช้ทราบว่าได้ทำการซ่อนข้อความเรียบร้อยแล้วจากนั้นโปรแกรมจะแสดงภาพเปรียบเทียบระหว่างภาพก่อนการซ่อนข้อความและภาพหลังจากซ่อนข้อความแล้วให้ผู้ใช้ดู

รูปที่ 3.17 แสดงข้อความการทำงานว่าได้ทำการซ่อนข้อมูลเรียบร้อยแล้ว

รูปที่ 3.18 แสดงภาพเปรียบเทียบระหว่างภาพก่อนการซ่อนข้อความและภาพหลังจากซ่อนข้อความ

รูปที่ 3.19 แสดงข้อความแจ้งข้อผิดพลาดในกรณีที่ภาพมีขนาดเล็กเกินกว่าที่จะนำข้อมูลไปซ่อนได้

รูปที่ 3.20 แสดงข้อความแจ้งข้อผิดพลาดในกรณีที่ผู้ใช้ยังไม่ได้เลือกรูปภาพเพื่อใช้ซ่อนข้อความ

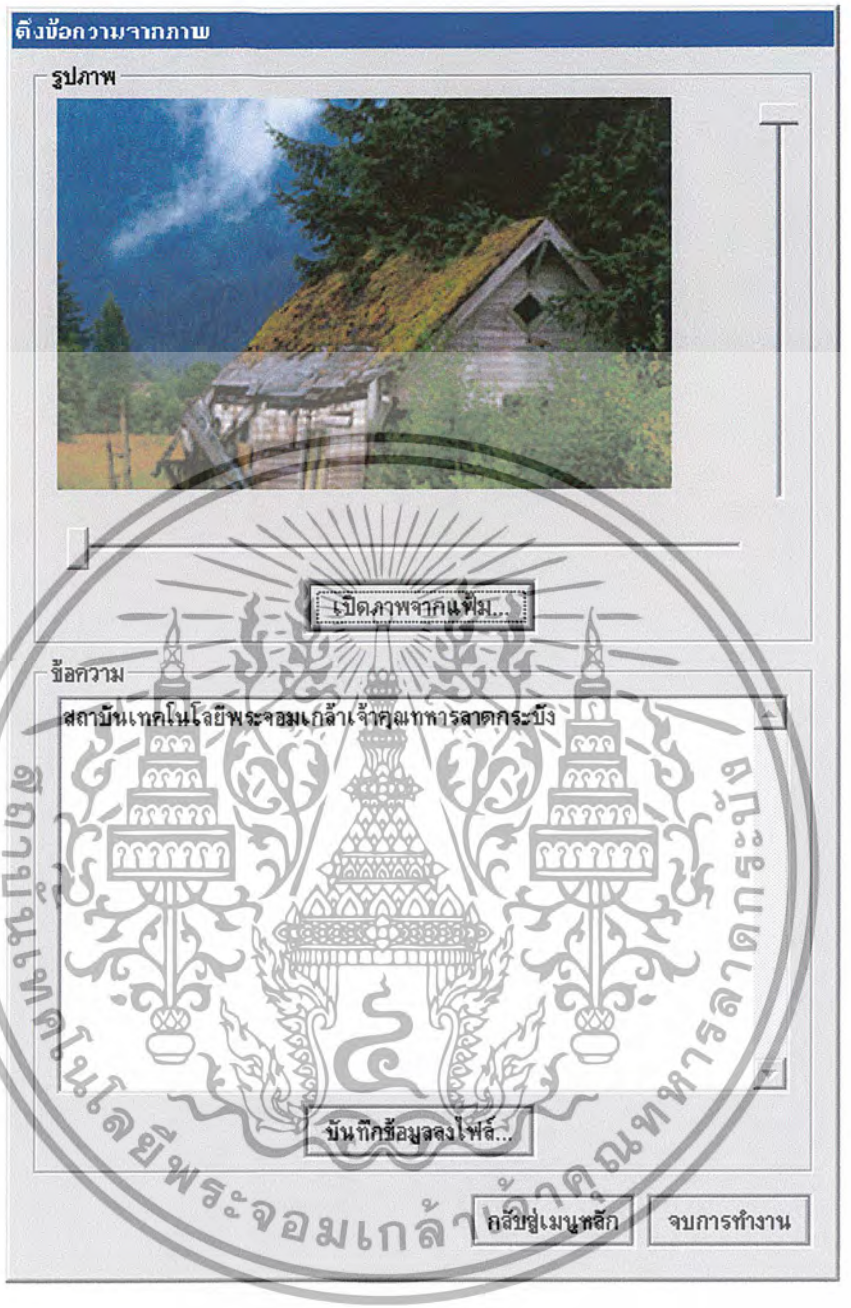
รูปที่ 3.21 แสดงข้อความแจ้งข้อผิดพลาดในกรณีที่ผู้ใช้ยังไม่ได้ป้อนหรือเลือกข้อความที่จะนำไปซ่อนลงในภาพ

หลังจากซ่อนข้อความเรียบร้อยแล้วให้ผู้ใช้กดปุ่ม กลับสู่เมนูหลัก เพื่อกลับไปเมนูหลัก หรือกดปุ่ม จบการทำงาน เพื่อออกจากโปรแกรม

3.5.2 ส่วนของการดึงข้อความออกจากภาพ

หน้าต่างการทำงานในส่วนนี้จะแบ่งการแสดงผลออกเป็น 2 ส่วน คือส่วนของภาพ และส่วนของข้อความที่ได้จากภาพ โดยการทำงานในส่วนนี้เริ่มจากการกดปุ่ม เปิดภาพจากแฟ้ม เพื่อเลือกภาพที่จะทำการดึงข้อความออกมา

รูปที่ 3.22 แสดงหน้าต่างส่วนของการดึงข้อความออกจากภาพ



รูป 3.23 แสดงตัวอย่างการดึงข้อความจากภาพ

จากรูปเราจะได้ว่าเมื่อกดปุ่มเปิดภาพจากแฟ้ม แล้วเลือกภาพเปิดขึ้นมานั้น ถ้าภาพดังกล่าวมีข้อความซ่อนอยู่ โปรแกรมจะแสดงข้อความนั้นในส่วนของข้อความ โดยจากตัวอย่างข้อความที่ซ่อนอยู่คือ “ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง ”

รูปที่ 3.24 แสดงข้อความแจ้งข้อผิดพลาดในกรณีที่ภาพที่เลือกเปิดขึ้นมาไม่มีการซ่อนข้อความอยู่

เมื่อเลือกภาพได้แล้วโปรแกรมจะทำการตรวจสอบว่าภาพที่เปิดขึ้นมานั้นมีการซ่อนข้อความเอาไว้หรือไม่ถ้ามีการซ่อนข้อความไว้โปรแกรมจะทำการดึงข้อความที่อยู่ในภาพออกมาแสดงในส่วนของข้อความจากนั้นผู้ใช้สามารถแก้ไขข้อความที่แสดงได้กดปุ่มบันทึกข้อมูลของไฟล์เพื่อบันทึกค่าข้อมูลที่ได้

หลังจากนั้น กดปุ่ม กลับสู่เมนูหลัก เพื่อกลับไปหน้าเมนูหลัก หรือกดปุ่ม จบการทำงาน เพื่อออกจากโปรแกรม

บทที่ 4
บทสรุปและข้อเสนอแนะ

4.1 การประเมินผล

โปรแกรมนี้เป็นโปรแกรมที่พัฒนาขึ้นเพื่อการซ่อนข้อความที่เป็นความลับ แล้วฝังลงไปในรูปภาพ เพื่อเพิ่มความปลอดภัยของข้อมูลระหว่างเครือข่ายคอมพิวเตอร์โดยภาพที่จะใช้ในการซ่อนข้อมูลนั้น จะต้องเป็นรูปภาพแบบ 24-bit bitmap หรือว่าเป็นภาพแบบ GIF เท่านั้น จึงจะสามารถใช้โปรแกรมช่วยซ่อนข้อความลงไปได้ โดยเมื่อผู้ใช้เลือกภาพที่ต้องการได้แล้ว ผู้ใช้ก็จะสามารถพิมพ์ข้อความที่ต้องการจะซ่อนลงไปได้ และจะนำข้อความดังกล่าว ไปซ่อนลงในภาพ โดยภาพที่จะเป็นตัวซ่อนข้อความนั้นต้องมีขนาดพื้นที่เพียงพอในการซ่อนข้อมูลลงไป

จากการทดลองใช้โปรแกรมทำให้พบว่าการที่ผู้ใช้เลือกภาพแล้วไม่สามารถจะซ่อนข้อความลงไปได้นั้น เกิดจากภาพที่ผู้ใช้เลือกขึ้นมานั้น เป็นภาพชนิดอื่นที่ไม่ใช่ .bmp แบบ 24-bit bitmap หรือ GIF ซึ่งจะเกิดขึ้นน้อยครั้งมาก

หลังจากที่ผู้ใช้ได้ทำการซ่อนข้อความลงในภาพแล้ว และได้ทำการบันทึกและส่งต่อแล้วนั้น ภาพดังกล่าวจะสามารถนำไปเปิดดูด้วยโปรแกรมอื่นๆ ได้ตามปกติได้เหมือนภาพทั่วๆ ไป โดยไม่ที่สามารถรับรู้ได้ว่ามีการซ่อนข้อความอยู่ในภาพนั้นๆแต่ถ้าผู้ใช้มีโปรแกรมนี้ก็จะสามารถตรวจสอบได้ว่าได้มีการซ่อนข้อความรวมอยู่ในภาพหรือไม่ และถ้ามีการซ่อนข้อความอยู่ในภาพก็จะสามารถเปิดอ่านข้อความนั้นๆได้ ถ้ามีการซ่อนข้อความมาด้วย

4.2 ข้อควรปรับปรุงแก้ไข

1. โปรแกรมนี้สามารถใช้ได้กับภาพ 24-bit bitmap , 8-bit bitmap และ GIF เท่านั้น
2. ในการซ่อนข้อความลงในภาพนั้นมีข้อจำกัดว่า สามารถใช้ได้กับภาษาไทย และ ภาษาอังกฤษเท่านั้น
3. ควรพัฒนาโปรแกรมให้สามารถใช้กับไฟล์ภาพในรูปแบบอื่นๆได้ด้วย หรือไม่ก็ควรที่จะเพิ่มที่ส่วนที่ใช้แปลงไฟล์ภาพต่างๆให้เป็นไฟล์รูปแบบที่ใช้ได้ในโปรแกรม
4. ควรพัฒนาส่วนของการบีบอัดข้อมูลให้มีประสิทธิภาพมากขึ้นกว่าเดิม
5. ในกรณีที่ภาพที่ใช้ซ่อนโปรแกรมเป็นภาพแบบ GIF โปรแกรมนี้จะ ไม่สามารถใช้ได้กับภาพ GIF แบบเคลื่อนไหวได้

4.3 สรุปผล

ผลที่ได้จากโปรแกรม โดยใช้เทคนิคและหลักการต่างๆที่ได้กล่าวมาแล้วในบทที่ 2 จะทำให้เราสามารถ สร้างโปรแกรมเพื่อซ่อนข้อความลงในไฟล์ภาพได้ดีขึ้น และภาพนั้น ก็ยังสามารถเปิดดูได้ตามปกติโดยที่ข้อความที่ซ่อนอยู่ในภาพจะไม่มีทางถูกแสดงออกมา ถ้าไม่ใช้โปรแกรมนี้ช่วยในการเปิด

โดยตัวโปรแกรมมีความสามารถต่างๆดังนี้

1. โปรแกรมสามารถซ่อนข้อความลงไปในรูปภาพได้ โดยที่ไม่ทำให้ภาพนั้นๆมีขนาดใหญ่ขึ้น(เฉพาะภาพ Bitmap) เพราะว่าโปรแกรมได้ซ่อนข้อความต่างๆ ลงในจุดสีของภาพ ซึ่งคุณภาพของภาพที่เปลี่ยนไปไม่มีผลกับสายตาของมนุษย์ นั่นคือไม่สามารถสังเกตเห็นสีที่เปลี่ยนไป

2. ก่อนที่จะซ่อนข้อความลงในภาพโปรแกรมจะทำการบีบอัดข้อความก่อนเพื่อลดขนาดเนื้อที่ที่ต้องการในการซ่อน ทำให้สามารถซ่อนข้อความได้ได้มากขึ้น

3. ภาพที่ผ่านการซ่อนข้อความด้วยโปรแกรมนี้ สามารถนำไปใช้เปิดกับโปรแกรมดูภาพทั่วไปได้ แต่จะไม่สามารถแสดงข้อความได้

โปรแกรมนี้ถูกพัฒนาขึ้นบน Window XP และต้องใช้กับภาพ 24- bit bitmap , 8-bit bitmap และ GIF มีการติดต่อกับผู้ใช้โดยใช้เมาส์

4.4 ข้อเสนอแนะ

1. โปรแกรมควรจะใช้ได้กับไฟล์รูปภาพชนิดอื่นๆได้นอกเหนือจาก 24 –bit bitmap หรือภาพ GIF ซึ่งจะต้องศึกษารูปแบบในการจัดเก็บไฟล์ภาพแต่ละรูปแบบ เพื่อจะได้นำไปศึกษาต่อว่าจะสามารถซ่อนข้อความลงไปในไฟล์ภาพเหล่านั้นได้อย่างไรโดยที่จะไม่ทำให้คุณภาพของภาพนั้นๆ เปลี่ยนแปลงไปจนสามารถสังเกตเห็นได้

2. ควรมีการปรับปรุงให้โปรแกรมสามารถใช้งานได้กับภาษาอื่นๆ นอกเหนือจากภาษาไทยและภาษาอังกฤษ

3. ควรจะปรับปรุงส่วนบีบอัดข้อมูลให้สามารถบีบอัดข้อมูลได้มีประสิทธิภาพมากยิ่งขึ้นเพื่อให้สามารถซ่อนข้อความที่มีขนาดยาวๆได้มากขึ้น โดยอาจจะทำเพิ่มส่วน Huffman model อื่นๆเข้าไปในโปรแกรม หรืออาจจะเปลี่ยนวิธีการบีบอัดข้อมูลที่มีประสิทธิภาพมากกว่านี้

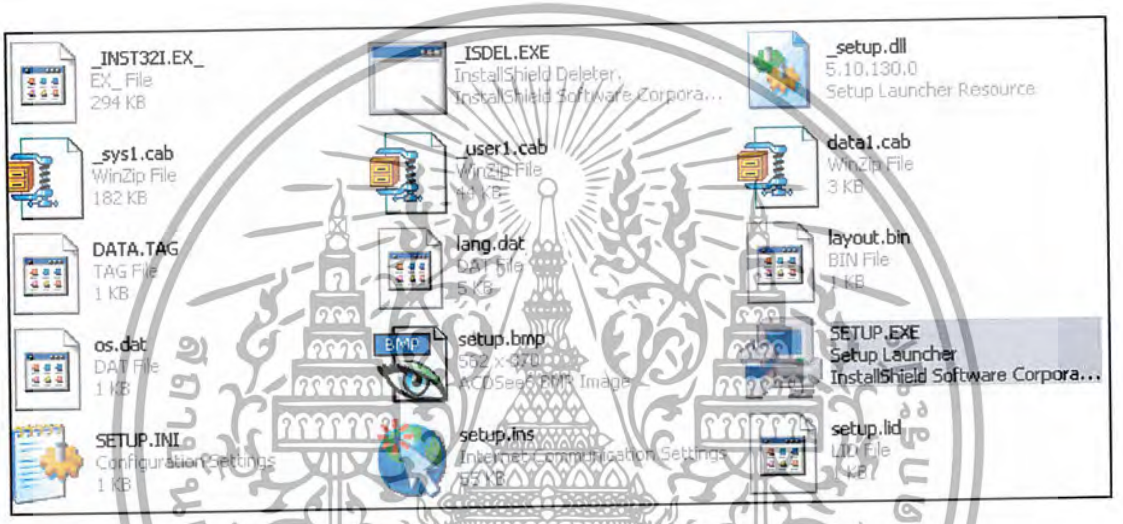
4. อาจจะพัฒนาโปรแกรมต่อโดยสามารถทำให้โปรแกรมสามารถซ่อนลงไปในไฟล์ข้อมูลของเสียงได้ด้วย

5. การที่จะทำการพัฒนาโปรแกรมนี้ต่อๆไป เพื่อเป็นการเพิ่มประสิทธิภาพของตัวโปรแกรมนั้นผู้พัฒนาควรที่จะต้องเน้นการศึกษาเกี่ยวกับเรื่อง Steganography ,Huffman Coding , Graphics File format เพิ่มเติม

ภาคผนวก

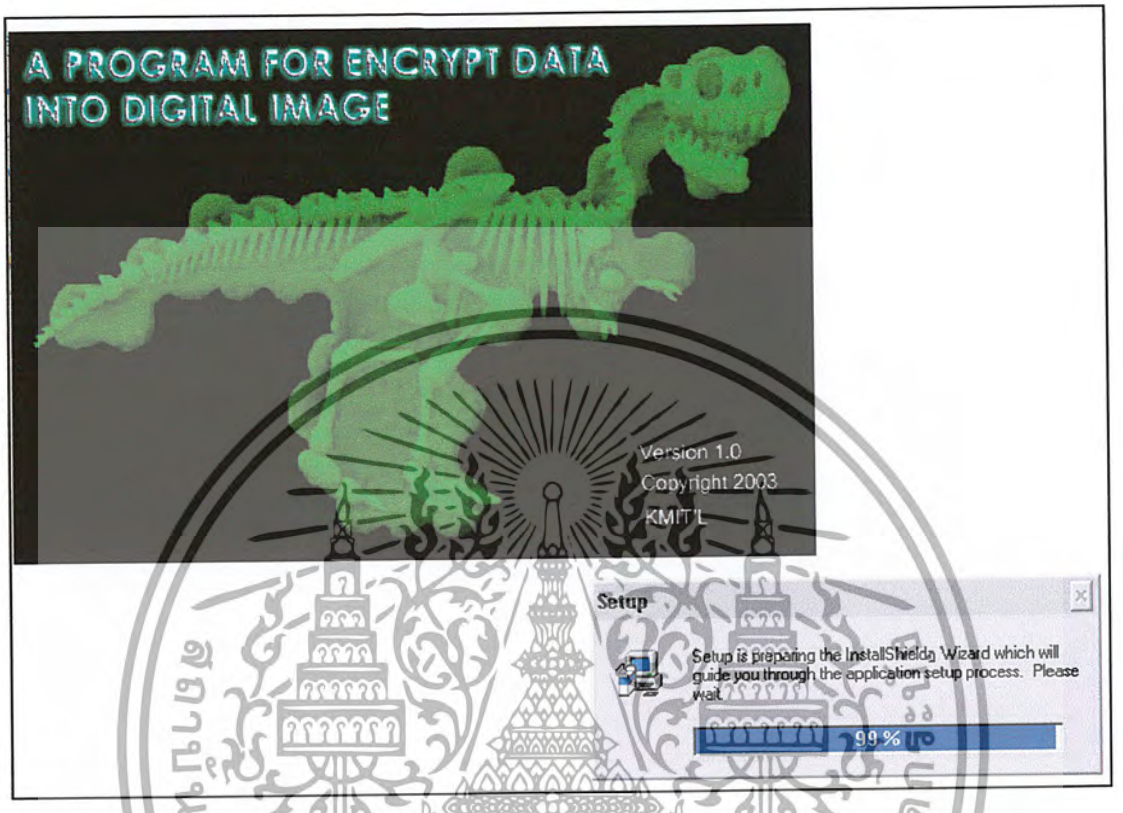
ภาคผนวก ก.
วิธีการติดตั้งโปรแกรม

ขั้นที่ 1 การติดตั้งโปรแกรมนั้นจะทำได้โดยการเปิดหน้าต่าง Window Explorer ขึ้นมา จากนั้นเลือกไปที่ Folder ที่ใช้เก็บไฟล์ที่ใช้ในการ Setup ซึ่งจะได้ดังรูปด้านบนนี้ จากนั้นทำการดับเบิลคลิกที่ SETUP.EXE เพื่อเข้าสู่ขั้นตอนการติดตั้ง



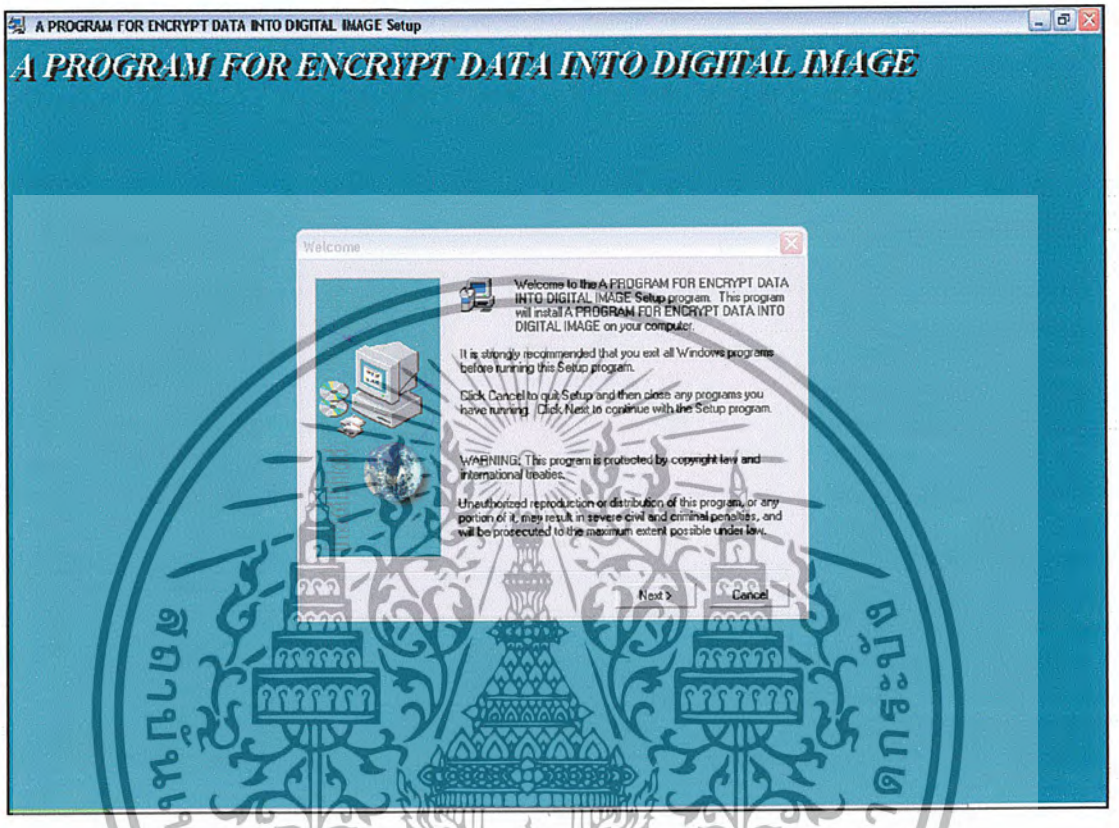
รูปที่ ก-1 ภาพแสดงรูปไฟล์ Setup

ขั้นที่ 2 หลังจาก ดับเบิลคลิกที่ SETUP.EXE โปรแกรมการติดตั้งจะแสดงชื่อโปรแกรมและแสดงหน้าจอให้เห็นว่ากำลังทำการติดตั้ง



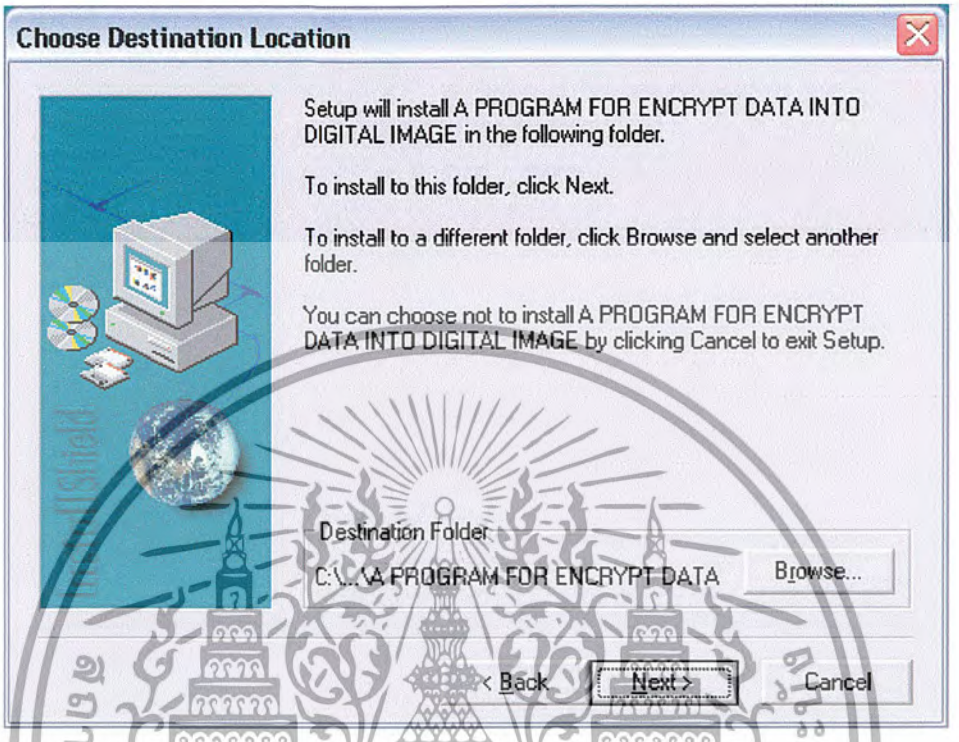
รูปที่ ก-2 ภาพแสดงการเตรียมการติดตั้งโปรแกรม

ขั้นที่ 3 เมื่อผ่านการแสดงการติดตั้งเรียบร้อยแล้ว จะเข้าสู่หน้าแสดงการติดตั้ง รายละเอียดของโปรแกรมนี้ จากนั้นทำการคลิกที่ปุ่ม Next เพื่อเข้าสู่ขั้นตอนต่อไป



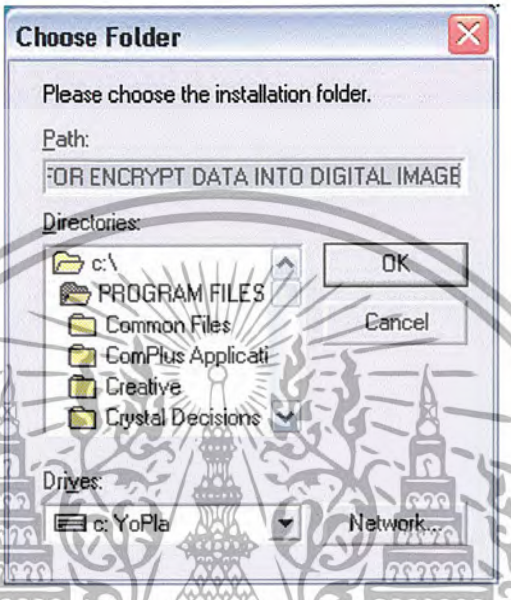
รูปที่ ก-3 ภาพแสดงหน้าจอการติดตั้งโปรแกรม

ขั้นที่ 4 โปรแกรมจะเริ่มเข้าสู่การติดตั้ง โดยผู้ใช้จะเลือกว่าต้องการที่จะติดตั้งโปรแกรมนี้เอาไว้ที่โฟล์เดอร์ไหน โดยการคลิกที่ปุ่ม Browse



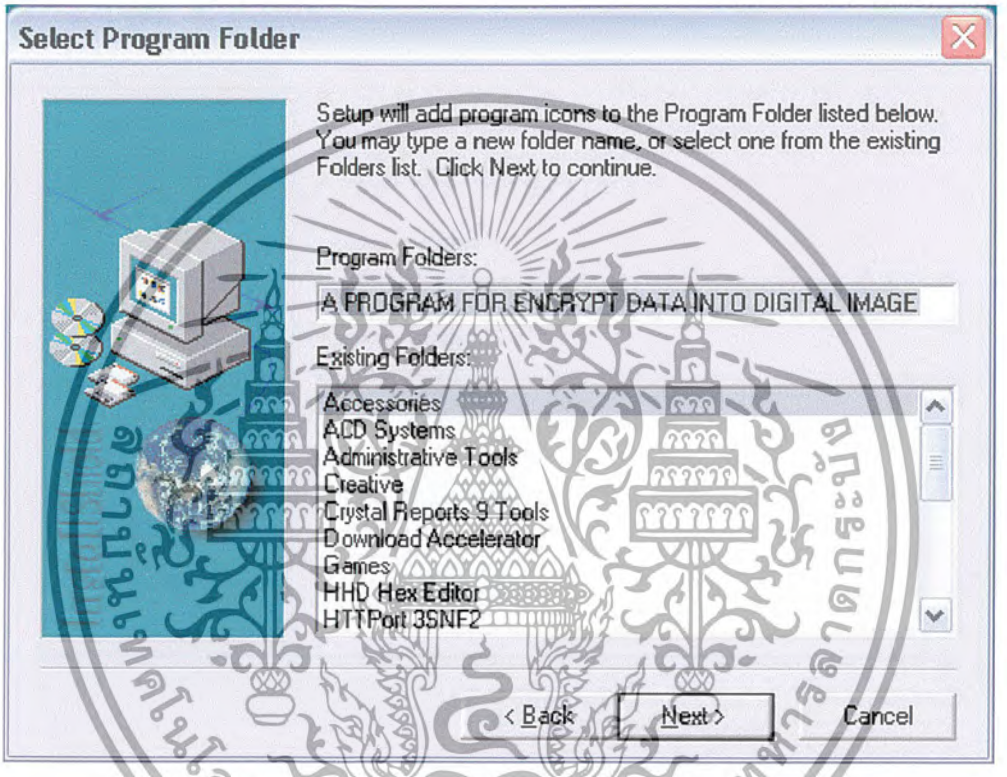
รูปที่ ก-4 ภาพแสดงการเลือกโฟล์เดอร์ที่จะเก็บโปรแกรม

ขั้นที่ 5 หลังจากการกดปุ่ม Browse จะแสดงหน้าต่างที่ให้ผู้ทำการติดตั้งเลือกว่าต้องการจะติดตั้งโปรแกรมไว้ยังที่ใด หลังจากเลือกเสร็จแล้วให้กด OK แลปุ่ม Next ตามลำดับ เพื่อดำเนินการในขั้นตอนต่อไป



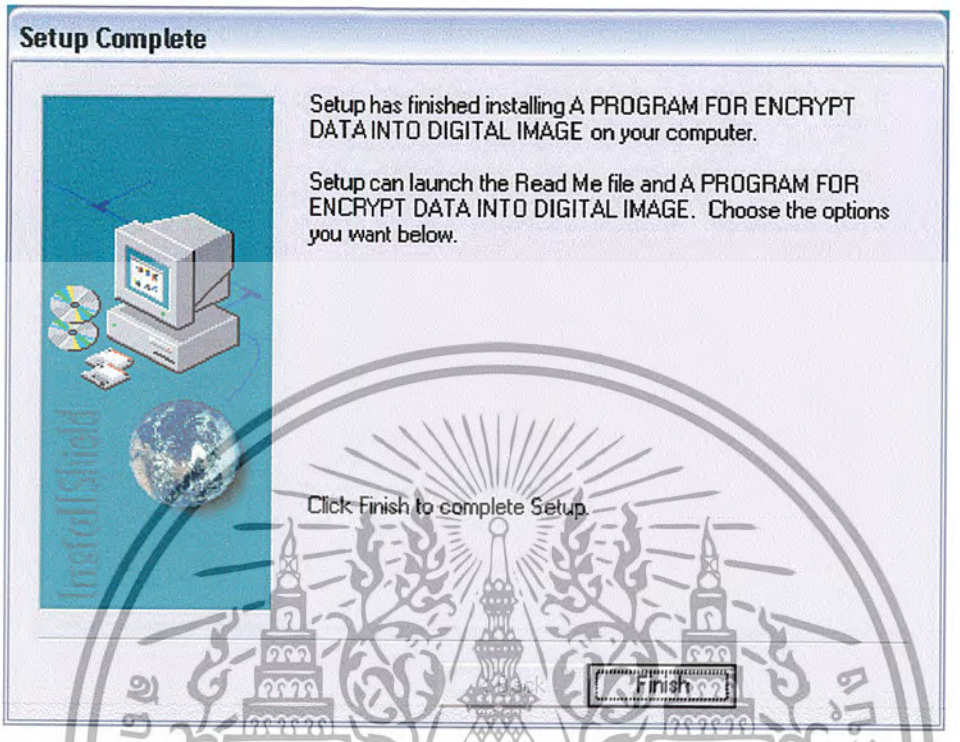
รูปที่ ก-5 ภาพแสดงการเลือกโฟลเดอร์เพื่อทำการติดตั้งโปรแกรม

ขั้นที่ 6 เมื่อผ่านขั้นตอนการเลือก Path ที่ต้องการจะติดตั้งโปรแกรมแล้ว จะเข้าสู่ขั้นตอนการกำหนดชื่อ Directory ที่จะใช้เรียกการใช้งานโปรแกรม (อยู่ในส่วนของการกด Start Menu --> Programs) โดยในช่อง Program Folders ให้ใส่ชื่อที่ต้องการ จากนั้นคลิกที่ปุ่ม Next เพื่อเข้าสู่ขั้นตอนต่อไป



รูปที่ ก-6 ภาพแสดงขั้นตอนการกำหนดชื่อ Directory

ขั้นที่ 7 ขั้นตอนสุดท้ายที่แสดงว่าโปรแกรมได้ถูกติดตั้งลงไปที่เครื่องเรียบร้อยแล้ว



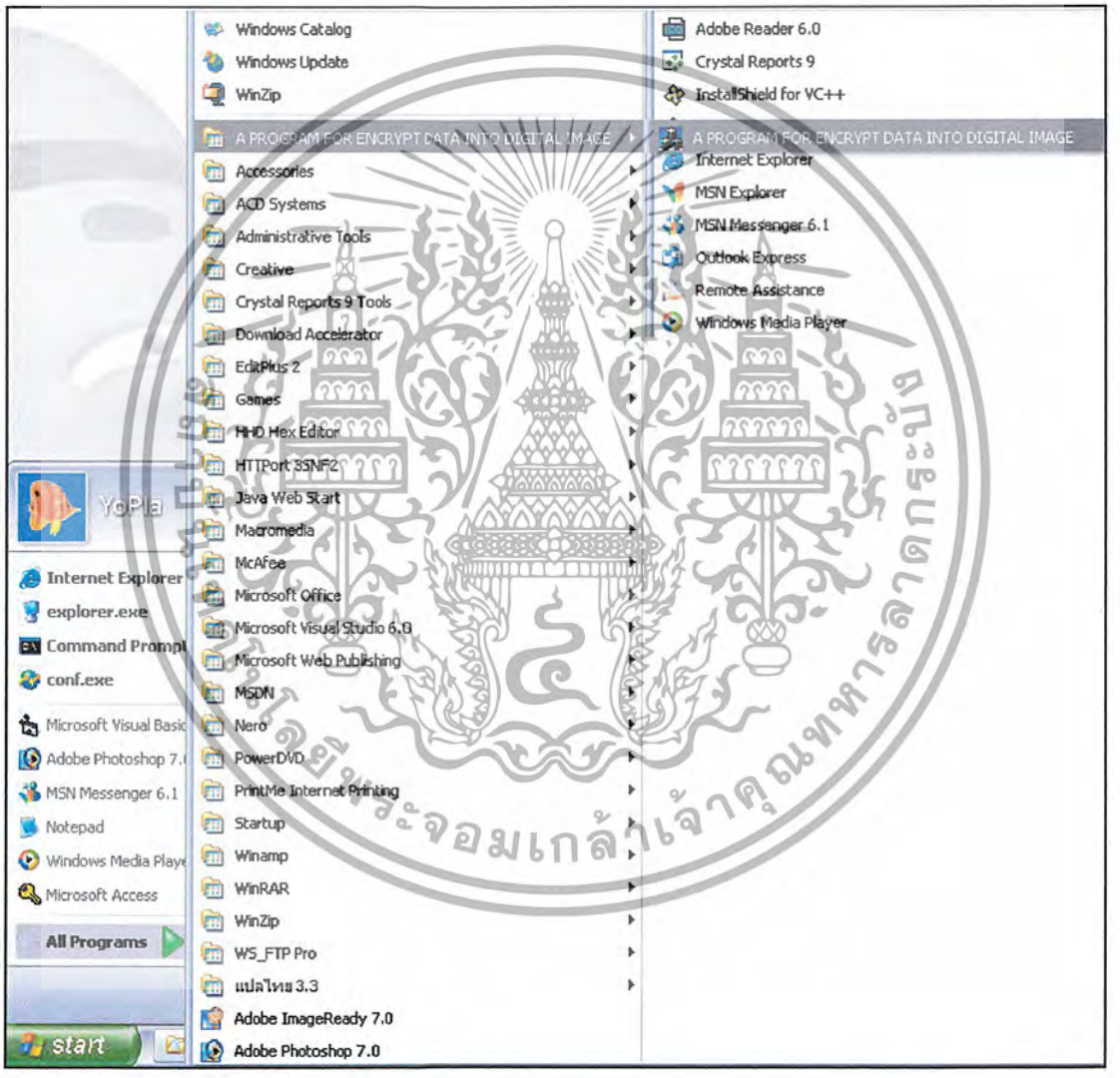
รูปที่ ก-7 ภาพแสดงว่าโปรแกรมได้ถูกติดตั้งเรียบร้อยแล้ว

ภาคผนวก ข.

วิธีการใช้งานโปรแกรม

การเรียกใช้โปรแกรมจากเมนู Start ของ Windows

การเรียกใช้งานทำได้โดยการ กดที่ Start Menu -> A PROGRAM FOR ENCRYPT DATA INTO DIGITAL IMAGE -> A PROGRAM FOR ENCRYPT DATA INTO DIGITAL IMAGE



รูปที่ ข-1 ภาพการเรียกโปรแกรมจากเมนู Start

ภาคผนวก ค.
ขั้นตอนวิธีที่ใช้ในโปรแกรม

ในส่วนนี้จะช่วยอธิบายในเรื่องการการใช้โปรแกรม ที่นำมาใช้ในโครงงานปัญหาพิเศษนี้ โดยการใช้ภาษา Microsoft Visual C ++ 6.0 ในการเขียนโปรแกรม โดยจะแบ่งเป็นขั้นตอนต่างๆ ดังต่อไปนี้

Function การซ่อนข้อความลงภาพ

```
BOOL CYAPDlg::FN_WriteDataToImage()
{
        CString TmpHuffmanEncode[256];
        int i,j,TmpLen,PtrImg=54;
        unsigned char ascii;
        char GYPHeader;// for get YPHeader.GetAt(i)

        for(i=0;i<SortOccurenceLen;i++)
                TmpHuffmanEncode[SortOccurence[i]]=HuffmanEncode[i];

        if((YPHeader.GetLength()+TextToHuffmanEncodeLen)>(ImageFileLen-54)){ // Check
For enough Space of Image For Insert data(Encoded)
                AfxMessageBox("ภาพมีขนาดไม่เพียงพอต่อการซ่อนข้อมูล");
                return false;
        }
//      Insert YP Header into ImageFile
        for(i=0;i<YPHeader.GetLength();i++){
                GYPHeader=YPHeader.GetAt(i);
                if(GYPHeader=='0'){
                        if(ImageFile[PtrImg]%2!=0){// Make From Odd to Even
                                ImageFile[PtrImg]--;
                        }
                }else{
```

```
                    if(ImageFile[PtrImg]%2==0){// Make From Even to Odd
                        ImageFile[PtrImg]++;
                    }
                }
                PtrImg++;
            }
//          Insert Text Data into ImageFile
            for(i=0;i<TextFileLen;i++){
                ascii=(unsigned char)TextFile[i];
                TmpLen=TmpHuffmanEncode[ascii].GetLength();
                for(j=0;j<TmpLen;j++){
                    GYPHeader=TmpHuffmanEncode[ascii].GetAt(j);
                    if(GYPHeader=='0'){
                        if(ImageFile[PtrImg]%2!=0){// Make From Odd to Even
                            ImageFile[PtrImg]--;
                        }
                    }else{
                        if(ImageFile[PtrImg]%2==0){// Make From Even to Odd
                            ImageFile[PtrImg]++;
                        }
                    }
                    PtrImg++;
                }
            }
            return FN_WriteImageFile();// Call Fn For write data to file
}
```

Function การบีบอัดข้อมูลแบบ Huffman

Function ส่วนนี้จะประกอบไปด้วย 2 Function ทำหน้าที่ บีบข้อมูลเป็น Huffman พร้อมกับสร้าง YPHeader

```
void CYAPDlg::FN_SortOccurence()
{
        int i,j,Tmp;
        for(i=0;i<256;i++)
                CntChar[i]=0;// Clear value of CntChar

        for(i=0;i<TextFileLen;i++){// Count number of a char
                CntChar[(unsigned char)TextFile[i]]++;
 }
        SortOccurenceLen=0;//Clear length of array
        for(i=0;i<256;i++){
                if(CntChar[i]>0){//choose only ascii to have occurence
                        SortOccurenceLen++;    //Count number of char Becaful
                }
        }
        delete [] SortOccurence;
        SortOccurence=new int [SortOccurenceLen];//allocate mem for array,it keep ascii of
chars
        j=0;
        for(i=0;i<256;i++){   // Set Ascii to Sort
                if(CntChar[i]>0){
                        SortOccurence[j]=i;
                        j++;
                }
        }
        for(i=0;i<SortOccurenceLen-1;i++){ //Sort Max to Min
```

```
                for(j=i+1;j<SortOccurenceLen;j++){
                        if(CntChar[SortOccurence[i]]>CntChar[SortOccurence[j]]){
                                Tmp=SortOccurence[i];
                                SortOccurence[i]=SortOccurence[j];
                                SortOccurence[j]=Tmp;
                        }
                }
        }
        FN_HuffmanEncode(SortOccurenceLen);//When Sort complete[we have
SortOccurenceLen] then call FN_HuffmanEncode for generate Huffman Code

void CYAPDlg::FN_HuffmanEncode(int GCntNumChar)
{
        struct Parent{
                int occurence;
                CString hand;
                Parent *head;
                Parent(){head=NULL;}
        };
        struct Leaf{
                int ascii;
                int occurence;
                CString hand;
                Parent *head;
        };
        Parent *pNow1,*pNow2,*Show;
        Leaf *CNode;
        int i,TmpOccurence=0;
        char buffer[8],buffer2[8],buffer3[24]; // For convert int to BINARY(String NAJA)
        CString ascii;     // For keep ASCII Code[Binary 8 bit Ex 10001010]

        delete [] HuffmanEncode;
```

```
HuffmanEncode=new CString [GCntNumChar];// Allocate HuffmanEncode for keep
Huffman Encode

switch(GCntNumChar){
case 1:
        HuffmanEncode[0]="1";
        break;
case 2:
        HuffmanEncode[0]="1";HuffmanEncode[1]="0";
        break;
default :
        CNode=new Leaf[GCntNumChar];
        CString Tmp1;
        for(i=0;i<GCntNumChar;i++){    // Set ascii & occurence value into Leaf  Node
                CNode[i].ascii=SortOccurence[i];
                CNode[i].occurence=CntChar[SortOccurence[i]];
        }
        pNow1=new Parent;
        pNow1->occurence=CNode[0].occurence+CNode[1].occurence;
        CNode[0].hand="1";      CNode[0].head=pNow1;
        CNode[1].hand="0";      CNode[1].head=pNow1;
        i=2;
        while(i<GCntNumChar){
                if(i==(GCntNumChar-1)){
                        pNow1->hand="1";
                        CNode[i].hand="0";
                        CNode[i].head=NULL;
                }else{
                        if(pNow1->occurence<CNode[i].occurence){
                                pNow1->hand="1";
                                CNode[i].hand="0";
                                pNow1->head=new Parent;
```

```
                    CNode[i].head=pNow1->head;
                    TmpOccurence=pNow1->occurence+CNode
[i].occurence;
                    pNow1=pNow1->head;
                    pNow1->occurence=TmpOccurence;
                }else{
                    pNow2=new Parent;
                    CNode[i].hand="1";
                    CNode[i++].head=pNow2;
                    CNode[i].hand="0";
                    CNode[i].head=pNow2;
                    pNow2->occurence=CNode[i].occurence+CNode[i-
1].occurence;
                    pNow1->hand="1";
                    pNow2->hand="0";
                    pNow2->head=new Parent;
                    pNow1->head=pNow2->head;
                    TmpOccurence=pNow1->occurence+pNow2->
occurence;
                    pNow1=pNow2->head;
                    pNow1->occurence=TmpOccurence;
                }
            }
            i++;
        }
        for(i=0;i<GCntNumChar;i++){    // read tree and set to HuffmanEncode[i]
            HuffmanEncode[i].Empty();
            HuffmanEncode[i]=CNode[i].hand;
            Show=CNode[i].head;
            while(Show){
                HuffmanEncode[i]=Show->hand+HuffmanEncode[i];
                Show=Show->head;
```

```
                }
            }
        }

        // ============== Begin Make YP HEADER ==============
        YPHeader="0101100101010000";          // add "YP" to YPHeader

        TextToHuffmanEncodeLen=0;
        for(i=0;i<SortOccurenceLen;i++){// Find length for allocate char *
TextToHuffmanEncode
            TextToHuffmanEncodeLen+=(CntChar[SortOccurence[i]]*HuffmanEncode
[i].GetLength());
        }
        _itoa(TextToHuffmanEncodeLen,buffer3,2);// Convert number of all Text covert by
huffman code to binary string
        if (strlen(buffer3)>24){
            AfxMessageBox("ข้อความมีความยาวเกินกว่าที่จะซ่อนได้");
            return;
        }
        ascii=buffer3;
        while(ascii.GetLength()<24){
            ascii="0"+ascii; // Fill "0" for 24 bit ex 101010001010100010101000
        }
        YPHeader+=ascii;          // add number of all char data[encoded by Huffman]

        _itoa(SortOccurenceLen,buffer,2);// Convert number of Difference Char to binary string
        ascii=buffer;
        while(ascii.GetLength()<8){
            ascii="0"+ascii; // Fill "0" for 8 bit ex 10101000
        }
        YPHeader+=ascii;          // add number of difference char to YPHeader
```

```
for(i=0;i<SortOccurenceLen;i++){// Create Huffman Table Lookup
        // Insert Ascii
        _itoa(SortOccurence[i],buffer2,2);
        ascii=buffer2;
        while(ascii.GetLength()<8){
                ascii="0"+ascii;
        }
        YPHeader=YPHeader+ascii;
        // Insert lenght of huffman code
        _itoa(HuffmanEncode[i].GetLength(),buffer2,2);
        ascii=buffer2;
        while(ascii.GetLength()<8){
                ascii="0"+ascii;
        }
        YPHeader=YPHeader+ascii;
        // Insert huffman code
        YPHeader=YPHeader+HuffmanEncode[i];
}
}
```

Function จับคู่สีในตารางสีของภาพแบบ 256 สี

Function นี้ทำหน้าที่ จับคู่สีในตารางสี

```
void CYAPDlg::FN_GIF_MapColorTable()
{
        int
NumberOfColor,TransparentIndex=256,i,j,PtrImg,TmpDiff,Diff,ParentIndex,CmpIndex;
        int DiffRGB=4; //        ค่าความแตกต่างระหว่างสีแม่กับสีที่เอามาเปรียบเทียบ

        for(i=0;i<256;++i){
                ColorMapTable[i]=256;  // Clear Map Table Index
        }

        NumberOfColor=(int)pow(2,(ImageFile[10]&0x07)+1);     // Find Number of color
        PtrImg=(NumberOfColor*3)+13; // set pointer to first color on pallete

        if((ImageFile[PtrImg]==33)&&(ImageFile[PtrImg+1]==-7)){//       ตรวจสอบว่ามี
LocalDescriptor Extension หรือไม่ ถ้ามีจะทำการตรวจค่า สีที่เป็น transparent
                //MessageBox("Have LocalDesc Extension");
                PtrImg+=3;
                if ((ImageFile[PtrImg]&0x01)==1) {      // ตรวจสอบว่ามี Transparent COlor
หรือไม่ ถ้ามีก็หา Index ของมัน
                        PtrImg+=3;
                        TransparentIndex=(unsigned char)ImageFile[PtrImg];
                        Tmp1.Format("Index of Transparent Color At : %d
",TransparentIndex);       MessageBox(Tmp1);
                }
        }
        for(i=0;i<NumberOfColor;++i){
                if(i!=TransparentIndex){
                        ParentIndex=13+(3*i);
                        Diff=(DiffRGB-1)*3+1;  // set max limit of Difference
```

```
if(i%2==0){     //      ถ้า Index ของสีเป็นคู่ ต้องการสีที่มา map เป็น
Index คี่
                        for(j=0;j<NumberOfColor/2;++j){
                                CmpIndex=13+(3*(j*2+1));
                                if (  (abs((unsigned char)ImageFile
[ParentIndex]-(unsigned char)ImageFile[CmpIndex])<DiffRGB)&&(abs((unsigned char)
ImageFile[ParentIndex+1]-(unsigned char)ImageFile[CmpIndex+1])<DiffRGB)&&(abs
((unsigned char)ImageFile[ParentIndex+2]-(unsigned char)ImageFile[CmpIndex+2])
<DiffRGB)&&(CmpIndex!=TransparentIndex)  ){
                                        TmpDiff=abs
((unsigned char)ImageFile[ParentIndex]-(unsigned char)ImageFile[CmpIndex]) + abs((unsigned
char)ImageFile[ParentIndex+1]-(unsigned char)ImageFile[CmpIndex+1])+ abs((unsigned char)
ImageFile[ParentIndex+2]-(unsigned char)ImageFile[CmpIndex+2]);
                                        if ( TmpDiff < Diff)
{
        Diff=TmpDiff;

        ColorMapTable[i]=(j*2+1);       // set index to map table
                                        }
                                }
                        }
}else{  //      ถ้า Index ของสีเป็นคี่ ต้องการสีที่มา map เป็น Index คู่
                        for(j=0;j<NumberOfColor/2;++j){
                                CmpIndex=13+(3*(j*2));
                                if (  (abs((unsigned char)ImageFile
[ParentIndex]-(unsigned char)ImageFile[CmpIndex])<DiffRGB)&&(abs((unsigned char)
ImageFile[ParentIndex+1]-(unsigned char)ImageFile[CmpIndex+1])<DiffRGB)&&(abs
((unsigned char)ImageFile[ParentIndex+2]-(unsigned char)ImageFile[CmpIndex+2])
<DiffRGB)&&(CmpIndex!=TransparentIndex)  ){
                                        TmpDiff=abs
((unsigned char)ImageFile[ParentIndex]-(unsigned char)ImageFile[CmpIndex]) + abs((unsigned
```

```
char)ImageFile[ParentIndex+1]-(unsigned char)ImageFile[CmpIndex+1])+ abs((unsigned char)
ImageFile[ParentIndex+2]-(unsigned char)ImageFile[CmpIndex+2]);

                                                    if ( TmpDiff < Diff)
{
        Diff=TmpDiff;
        ColorMapTable[i]=j*2;  // set index to map table
                                                                }
                                                            }
                                                        }
                                                    }
                                                }
                                            }
}
```

บรรณานุกรม

รองศาสตราจารย์ธีรวัฒน์ ประกอบผล.2545.**การโปรแกรมภาษาซีสำหรับงานวิทยาศาสตร์.**พิมพ์ครั้งที่ 1.บริษัทดวงกมลสมัย จำกัด

ยุทธนา ลีลาศวัฒนกุล และ สัจจะ จรัสรุ่งวีวร.**2544.คู่มือการเขียนโปรแกรมและใช้งาน Visual C++ 6.0 ฉบับโปรแกรมเมอร์.**พิมพ์ครั้งที่1.กรุงเทพฯ:อินโฟเพส

พิรุธ อำนวยศิลป์.2544.**คู่มือการเขียนโปรแกรม Microsoft Visual C++ ฉบับเพื่อการใช้งานจริง.** พิมพ์ครั้งที่ 4. กรุงเทพฯ:บริษัท ซัคเซส มีเดีย จำกัด

รองศาสตราจารย์มัณฑนา ปราการสมุทร.2534.**การเขียนชุดคำสั่งภาษาซี** พิมพ์ครั้งที่ 3.กรุงเทพฯ: บริษัท ดวงกมล สมัย จำกัด

C.Wayne Brown and Barry J.Shepherd.1995.**Graphics File Formats reference and guide.** 1st **edition**.Manning Publication Co.

Steve Rimmer.1993.**Windows Bitmapped Graphics.** 1 st edition .Windcrost Books

Mark Nelson.**LZW Data Compression.**[Online].Available:
http://www.dogma.net/markn/articles/lzw/lzw.htm

Karl Papadantonakis.**GIF file example.**[Online].Available:
http://www.async.caltech.edu/~kp/Misc/gifEx.html