บทที่ 6

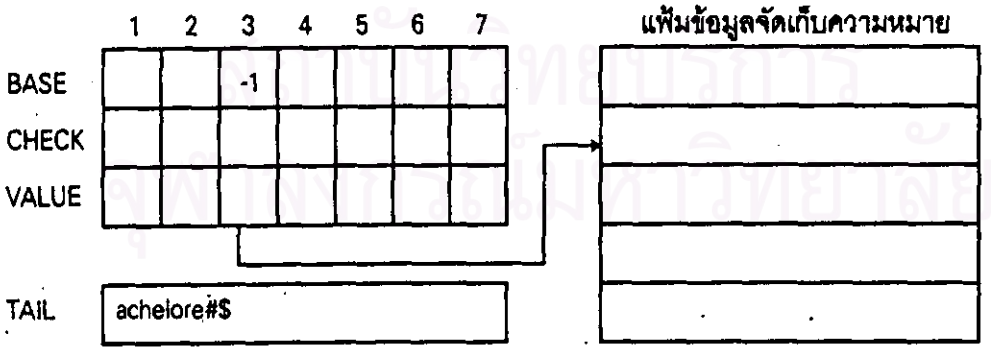
การออกแบบและพัฒนาพจนานุกรมอิเล็กทรอนิกส์

บทนี้จะกล่าวถึงขั้นตอนการออกแบบและพัฒนาพจนานุกรมอิเล็กทรอนิกส์ในเชิงวัตถุ ซึ่งจะกล่าวถึงคลาส ความสัมพันธ์ระหว่างคลาสและบริการของแต่ละคลาส

สำหรับงานวิจัยนี้ได้พัฒนาโปรแกรมสำหรับการบันทึก แก้ไข ลบและสืบค้นพจนานุกรมอิเล็กทรอนิกส์และเนื่องจากพจนานุกรมต้องการเนื้อที่ในการจัดเก็บมาก จึงทำการบีบอัดข้อมูลพจนานุกรมเพื่อลดเนื้อที่ในการจัดเก็บ ในส่วนของโครงสร้างการจัดเก็บข้อมูลได้แบ่งออกเป็น 2 รูปแบบ คือ

1. พจนานุกรมกำหนดโครงสร้างข้อมูลในส่วนของภาคแสดงความหมายให้
2. ผู้ใช้สามารถกำหนดโครงสร้างข้อมูลในส่วนของภาคแสดงความหมายเองได้

พจนานุกรมทั้งสองรูปแบบ ใช้โครงสร้างการจัดเก็บและสืบค้นข้อมูลแบบ ดี-เอส ทรี ซึ่งคิดค้นโดยนักวิจัยชาวญี่ปุ่น ชื่อ Aoe และสำหรับงานวิจัยนี้ได้นำการพัฒนาโปรแกรมของ น.ส.สมปรารถนา รัตยานนท์ มาเป็นต้นแบบ สำหรับการสืบค้นและเพิ่มคำศัพท์ลงในดี-เอส ทรี และได้เพิ่มเติมในส่วนของการแสดงความหมายและการค้นหาแบบระบุเฉพาะส่วนต้นของคำศัพท์ ในส่วนของการสืบค้นความหมายได้เพิ่มอะเรย์ value เพื่อจัดเก็บหมายเลขตรรชนีชี้ไปยังระเบียนที่ต้องการ ดังรูปที่ 6-1



รูปที่ 6-1 โครงสร้างการจัดเก็บข้อมูลของพจนานุกรมอิเล็กทรอนิกส์

ส่วนในการบีบอัดข้อมูลนั้นได้ใช้วิธีการบีบอัดข้อมูลแบบแอลแซดดับบลิว ซึ่งได้ใช้แนวทางการพัฒนาโปรแกรมของเนลสัน และเกลลี มาเป็นต้นแบบ โดยในตอนเริ่มแรกจะใช้ขนาดของบิต 9 บิตและเมื่อแฟ้มข้อมูลมีขนาดใหญ่ขึ้นก็จะเพิ่มเป็น 10 , 11, 12 จนกระทั่ง 15 บิต ตามขนาดของแฟ้มข้อมูล

การวิเคราะห์และออกแบบเชิงวัตถุ

ขั้นตอนการวิเคราะห์และออกแบบเชิงวัตถุ (Coad , 1993)

1. กำหนดวัตถุประสงค์และคุณลักษณะของระบบ (Identifying systems purpose and features)
2. ออกแบบจำลองเชิงวัตถุของระบบ (Object - Oriented Model) โดย
   - 2.1 กำหนดวัตถุ ซึ่งเป็นสิ่งที่สนใจของระบบ
   - 2.2 สร้างความสัมพันธ์ระหว่างวัตถุ
   - 2.3 กำหนดซีนาริโอ (scenarios) ซึ่งเป็นการแสดงการดำเนินการของวัตถุต่างๆ เมื่อมีการร้องขอบริการระหว่างวัตถุ เพื่อให้บรรลุวัตถุประสงค์หนึ่งๆ

1. กำหนดวัตถุประสงค์และคุณลักษณะของระบบ

วัตถุประสงค์ของระบบนี้ คือ สร้างพจนานุกรม อังกฤษ-ไทย สำหรับค้นหาความหมายของคำศัพท์ที่ต้องการ

คุณลักษณะของระบบ คือ สามารถเพิ่ม แก้ไข ลบข้อมูลและสืบค้นคำศัพท์และความหมายในฐานข้อมูลได้ และสามารถใช้งานร่วมกับโปรแกรมจุฬาจารึกหรือโปรแกรมประมวลผลคำอื่น

2. ออกแบบจำลองเชิงวัตถุของระบบ (Object Model)

มีผู้คิดค้นสัญลักษณ์สำหรับ ใช้แสดงแบบจำลองเชิงวัตถุไว้หลายท่านด้วยกัน เช่น Booch and Rumbaugh, Coad's notation ในที่นี้จะใช้ Coad's notation เพื่อแสดงแบบจำลองเชิงวัตถุสำหรับงานวิจัยนี้

แบบจำลองเชิงวัตถุประกอบด้วยส่วนสำคัญ 6 ส่วน คือ

2.1 วัตถุ (Objects) จากการพิจารณาวัตถุประสงค์และคุณลักษณะของระบบจะสามารถกำหนดวัตถุที่เป็นส่วนประกอบของระบบ ซึ่งในระบบนี้สามารถกำหนดวัตถุได้ ดังนี้

CContent เป็นคลาสที่ใช้จัดเก็บคำศัพท์ใน ดี-เอส ทรี

CLzw เป็นคลาสที่ใช้สำหรับการบีบอัดและขยายข้อมูล

CFixedLenRecDoc เป็นคลาสที่ใช้จัดเก็บและสืบค้นความหมายในแฟ้มข้อมูล

CChkBookDoc เป็นคลาสที่สืบทอดมาจากคลาส CFixedLenRecDoc

CCompDoc เป็นคลาสที่สืบทอดมาจากคลาส CFixedLenRecDoc

CDefDoc เป็นคลาสที่สืบทอดมาจาก CFixedLenRecDoc ใช้สำหรับจัดเก็บและสืบค้นความหมาย ในกรณีที่ผู้ใช้งานกำหนดโครงสร้างชนิดข้อมูลเอง

CCheckView เป็นคลาสที่สืบทอดมาจาก CFormView เป็นส่วนที่ติดต่อกับผู้ใช้งาน โดยใช้หรับการเพิ่มและแก้ไขคำศัพท์และความหมาย

CBookView เป็นคลาสที่สืบทอดมาจาก CRowView ใช้แสดงข้อมูลคำศัพท์และความหมายทั้งหมดในแฟ้มข้อมูล

CDefView เป็นคลาสที่สืบทอดมาจาก CFormView เป็นส่วนที่ติดต่อกับผู้ใช้งาน โดยสามารถบันทึกคำศัพท์และความหมายจากคลาสนี้ ในกรณีที่ผู้ใช้งานกำหนดโครงสร้างชนิดข้อมูลเอง

CFindDlg เป็นคลาสที่สืบทอดมาจาก CDialog เป็นส่วนที่ใช้สำหรับติดต่อกับผู้ใช้งาน โดยเมื่อผู้ใช้งานป้อนคำศัพท์ที่ต้องการค้นหา คลาสนี้จะนำความหมายของคำศัพท์นั้นๆ มาแสดง และหากค้นหาคำศัพท์แล้วไม่พบก็จะแสดงข้อความให้ทราบ

CAddDlg เป็นคลาสที่สืบทอดมาจาก CDialog เป็นส่วนที่ใช้สำหรับติดต่อกับผู้ใช้งาน โดยให้ผู้ใช้เลือกชนิดข้อมูลที่ต้องการ สำหรับกรณีที่ต้องการกำหนดโครงสร้างชนิดข้อมูลเอง

Chook เป็นคลาสที่ดำเนินการเชื่อมต่อพจนานุกรมกับจุฬาจารึก

2.2 ลักษณะประจำ (Object Attributes) หมายถึง สิ่งที่แสดงถึงคุณลักษณะหรือข้อมูลของวัตถุ

2.3 บริการของวัตถุ (object Services) หมายถึง ขบวนการซึ่งดำเนินการโดยวัตถุ เมื่อได้รับข่าวสาร

(Messages) ให้ดำเนินการ

2.4 Class Hierachies หมายถึง การจัดลำดับชั้นของคลาสต่างๆ (Yourdon, 1997)

ซึ่งแบ่งออกได้ 2 ประเภท คือ

2.4.1 generalization - specialization connection

เป็นการจัดลำดับชั้นของคลาสในลักษณะ parent-child หรือ superclass-subclass เช่น คลาสของลูกค้า เป็น superclass โดยมี คลาสของลูกค้าที่จ่ายเงินสดและคลาสของลูกค้าที่จ่ายเช็คเป็น subclass

2.4.2 whole - part connection

เป็นการจัดลำดับชั้นของคลาสในลักษณะ ที่มีคลาสหรือวัตถุเป็นส่วนประกอบหรือเป็นสมาชิกของอีกคลาสหนึ่ง เช่น คลาสของรถยนต์ จะมีคลาสของเครื่องยนต์ พวงมาลัย และประตู เป็นส่วนประกอบ

2.5 Object Connection หมายถึง สิ่งที่เชื่อมความสัมพันธ์ระหว่างคลาสหรือวัตถุ (Norman , 1995)

ซึ่งสามารถมีความสัมพันธ์ได้หลายแบบ เช่น 1:1, 1:N, N:N

2.6 message เป็นการร้องขอบริการระหว่างวัตถุ

ส่วนประกอบของแบบจำลองเชิงวัตถุ โดยวิธีการของโคด

object model component หมายถึง กลุ่มของคลาส ซึ่งแบ่งออกเป็น 4 กลุ่ม ดังนี้

1. Problem Domain (PD) ประกอบด้วยวัตถุที่เป็นสิ่งที่สนใจหรือเป็นปัญหาของระบบ
2. Human Interaction (HI) ประกอบด้วยวัตถุที่เป็นส่วนเชื่อมความสัมพันธ์ระหว่างวัตถุใน PD กับ ผู้ใช้งาน
3. Data Management (DM) ประกอบด้วยวัตถุที่เป็นส่วนเชื่อมความสัมพันธ์ระหว่าง PD และแฟ้มข้อมูล
4. System Interaction(SI) ประกอบด้วยวัตถุที่เป็นส่วนเชื่อมความสัมพันธ์ระหว่าง วัตถุใน PD และระบบอื่นหรืออุปกรณ์ต่างๆ

ลำดับที่	คลาส	ลักษณะประจำ	บริการของวัตถุ	ความสัมพันธ์กับคลาสอื่น
1	CFixLenRecDoc	m_header	OnReadExtraHeader OnWriteExtraHeader WriteHeader ReadHeader CreateNewRecord GetRecordCount UpdateRecord UpdateAllViewsWithRecord OnOpenDocument DeleteContents FileSeekRecord	CChkBookDoc CCompDoc
2	CChkBookDoc	m_extraHeader m_record m_nActiveRecord fcomp	OnSaveDocument SaveModified OnCreateNewRecord GetCheck UpdateCheck ChangeSelectionNextCheckNo ChangeSelectionToCheckNo GetActiveCheckNo GetFirstCheckNo GetLastCheckNo UpdateInitFileWithDocPath MaybeCommitDirtyCheck PackRecord ParseRecord CheckNoToRecordIndex RecordIndexToCheckNo GetActiveCheckno GetFirstCheckNo GetLastCheckNo NewCheck OnNextCheck OnUpdateNextCheck OnPrevCheck OnUpdatePrevCheck	CFixLenRecDoc CContent
3	CBookView		OnUpdate GetRowWidthHeight	CChkBookDoc

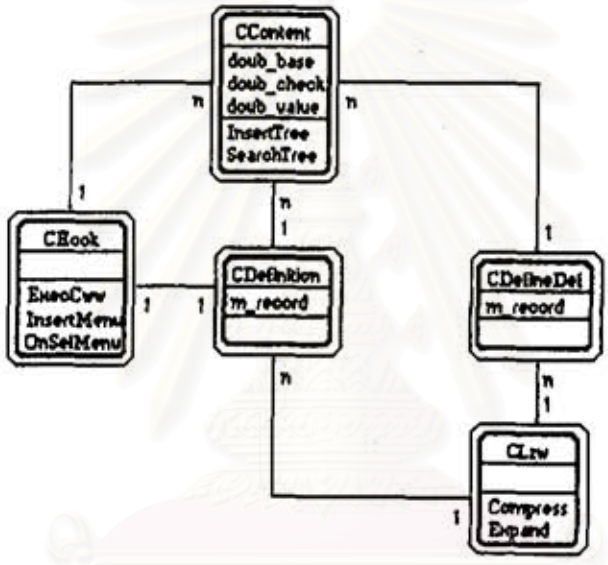
ลำดับที่	คลาส	ลักษณะประจำ	บริการของวัตถุ	ความสัมพันธ์กับคลาสอื่น
			GetActiveRow GetRowCount OnDrawRow ChangeSelectionNextRow ChangeSelectionToRow	
4	CCheckView	m_nCheckNo m_strWord m_strMean m_strSam m_strSyno m_strAnto	OnUpdate MaybeCommitDirtyCheck OnEditCommitCheck OnPaint	CChkBookDoc
5	CCompDoc	m_header m_extraHeader m_record	OnCreateNewRecord WriteHeader OnOpenFile OnSaveFile PackRecord ParseRecord	CFixLenRecDoc CLzw
6	CContent	doub_base doub_check doub_value POS	wordtoint ctoint Rd_Base Rd_Check Rd_Value A_Insert Set_List Insert_Str Str_Tail Modify X_Check B_Insert Init_Rtn Push_Base Push_Check Rd_Array Wr_Array Fetch_Str Str_Cmp	CChkBookDoc
7	CDefDoc	def_array	OnCreateNewRecord	CDefView

ลำดับที่	คลาส	ลักษณะประจำ	บริการของวัตถุ	ความสัมพันธ์กับคลาสอื่น
		m_nActiveRecord	WriteHeader	CContent
		m_file	OnReadExtraHeader	
		deffile	OnWriteExtraHeader	
		m_header	UpdateRecord	
		m_extraHeader	FileSeekRecord	
		tab	ReadHeader	
		maxtype	OnNewRecord	
		maxpron	MaybeCommitDirtyCheck	
		maxdef	PackRecord	
		maxsyno	ParseRecord	
		maxanto	UpdateChkBox	
		maxabb	GetChkBox	
		maxsam	GetActiveCheckNo	
		maxdate	ChangeSelectionNextCheckNo	
			CheckNoToRecordIndex	
			RecordIndexToCheckNo	
			UpdateCheck	
			NewRecord	
			OpenDefFile	
			SaveDefFile	
			InsertDef	
			GetLastCheckNo	
			GetFirstCheckNo	
			CreateDefFile	
			ChangeSelectionToCheckNo	
			RecordCount	
			PrevCheckNo	
			NextCheckNo	
8	CDevView	maxtype	OnUpdate	CDefDoc
		maxpron	MaybeCommitDirtyCheck	
		maxdef	OnCreate	
		maxsyno	OnCommit	
		maxanto	OnNew	
		maxabb	OnSaveFile	
		maxsam	OnDefInsert	
		maxdate	OnNext	
		m_nCheckNo	OnPrev	

ลำดับที่	คลาส	ลักษณะประจำ	บริการของวัตถุ	ความสัมพันธ์กับคลาสอื่น
		m_editword m_edittype m_editpron m_editdef m_editsyno m_editanto m_editabb m_editsam m_editdate m_stword m_sttype m_stpron m_stdef m_stsyno m_stanto m_stabb m_stsam m_stdate m_nActiveRecord		
9	CFindDlg		OnButSearch OnButPaste OnButCopy OnButUpdate OnButDelete	CContent CCheckBookDoc
10	CAddDlg	dmaxtype dmaxpron dmaxdef dmaxsyno dmaxanto dmaxabb dmaxsam dmaxdate	OnOK	CDefDoc CDefView
11	CLzw	dict decode_stack next_code current_code_bits next_bump_code	InitializeDictionary InitializeStorage CompressFile ExpandFile Find_child_node	CCompDoc

ลำดับที่	คลาส	ลักษณะประจำ	บริการของวัตถุ	ความสัมพันธ์กับคลาสอื่น
			decode_string	
12	Chook		ExecCww InsertMenu OnSearch	Cww78

ตารางที่ 6-1 ส่วนประกอบในแบบจำลองเชิงวัตถุของพจนานุกรมอิเล็กทรอนิกส์



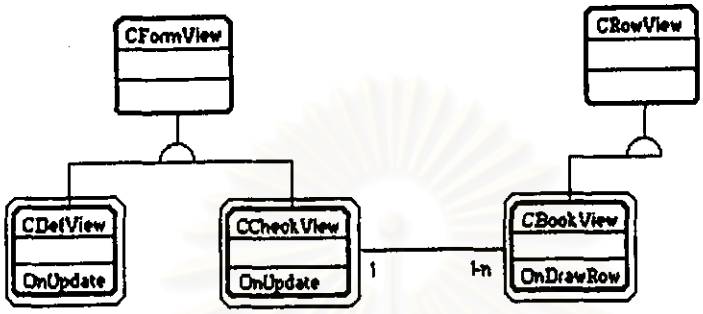
รูปที่ 6-2 แบบจำลองเชิงวัตถุของพจนานุกรมอิเล็กทรอนิกส์ (PD)

จากรูปที่ 6-2 คลาส CDefinition ดำเนินการเกี่ยวกับการจัดเก็บข้อมูลคำศัพท์และความหมาย ส่วนคลาส CContent ดำเนินการในการสร้างดีเอส-ทรี ซึ่งจะนำคำศัพท์จากคลาส CDefinition มาสร้างทรี โดยใช้บริการ InsertTree ในคำศัพท์หนึ่งคำจะประกอบด้วยหลายโหนด นอกจากนี้ในการสืบค้นคำศัพท์ คลาส CContent จะใช้บริการ SearchTree เมื่อพบคำศัพท์ที่ต้องการจะให้ดรรชนีชี้ไปยังตำแหน่งระเบียนในแฟ้มข้อมูล

ส่วนคลาส CDefineDef จะดำเนินการเช่นเดียวกับคลาส CDefinition แต่จะดำเนินการกับแฟ้มข้อมูลที่ผู้ใช้งานกำหนดโครงสร้างในภาคแสดงความหมายเอง

คลาส CDefinition และคลาส CDefineDef สัมพันธ์กับคลาส CLzw โดยที่ใช้บริการ Compress ลดขนาดข้อมูลของ CDefinition และ CDefineDef ในขณะที่ไม่ได้ใช้งานพจนานุกรม และขยายกลับโดยใช้บริการ Expand เมื่อต้องการใช้งาน

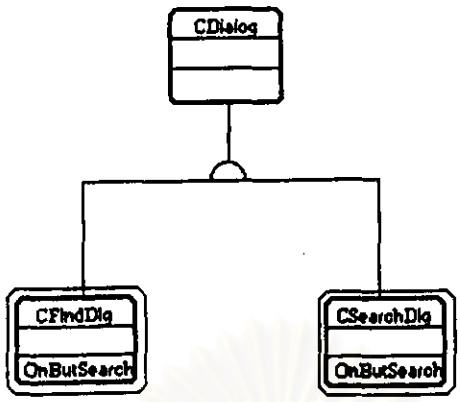
คลาส CHook ดำเนินการในการเชื่อมต่อกับจุฬาจารึก 78 ซึ่งเมื่อผู้ใช้จุฬาจารึกเรียกใช้พจนานุกรม คลาส CHook จะร้องขอบริการจาก คลาส CContent เพื่อสืบค้นคำศัพท์ในดีเอส-ทรี และคลาส CDefinition เพื่อนำความหมายมาแสดง



รูปที่ 6-3 การบันทึกคำศัพท์และความหมาย (HI)

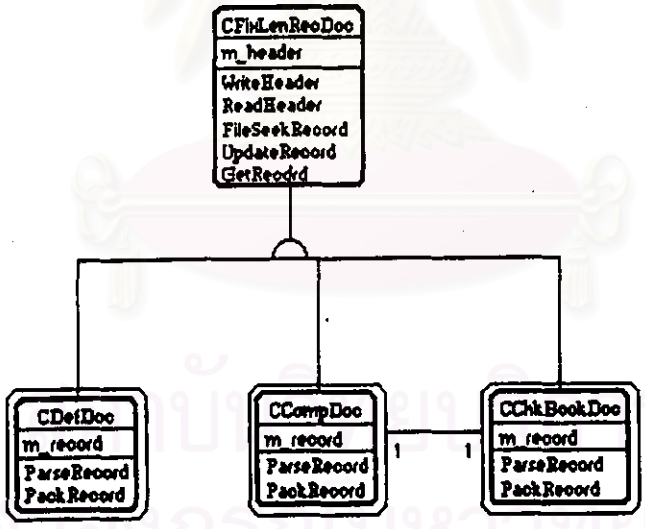
จากรูปที่ 6-3 คลาส CDefView และคลาส CCheckView สืบทอดมาจากคลาส CFormView ซึ่งดำเนินการในการบันทึก และแก้ไขข้อมูลคำศัพท์และความหมาย คลาสทั้งสองจะใช้บริการ OnUpdate ปรับปรุงการแสดงผลทุกครั้งที่ข้อมูลในแฟ้มข้อมูลเปลี่ยนแปลง คลาส CCheckView ดำเนินการกับข้อมูลที่พจนานุกรมกำหนดโครงสร้างภาคแสดงความหมายมาให้ ส่วนคลาส CDefView ดำเนินการกับข้อมูลที่ผู้ใช้กำหนดโครงสร้างข้อมูลเอง

คลาส CBookView สืบทอดมาจากคลาส CRowView ซึ่งดำเนินการในการแสดงข้อมูลทีละหลายระเบียน คลาส CBookView สัมพันธ์กับคลาส CCheckView โดยที่ในหนึ่งจอภาพ คลาส CCheckView จะแสดงข้อมูลทีละ 1 ระเบียน ส่วนคลาส CBookView จะแสดงข้อมูลทีละหลายระเบียนในลักษณะเป็นแถว โดยสามารถเลื่อนจอภาพขึ้นลงได้



รูปที่ 6-4 การสืบค้นความหมาย (HI)

รูปที่ 6-4 คลาส CFindDlg และคลาส CSearchDlg สืบทอดมาจาก คลาส CDialog ทั้งสองคลาสนี้ดำเนินการในการรับคำศัพท์ที่ต้องการสืบค้นความหมายจากผู้ใช้งาน และบริการ OnButSearch มีหน้าที่ในการนำความหมายของคำศัพท์ที่ต้องการมาแสดง

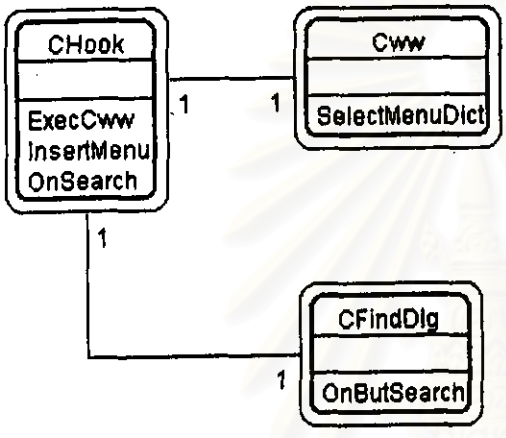


รูปที่ 6-5 การจัดการแฟ้มข้อมูลพจนานุกรม (DM)

จากรูปที่ 6-5 คลาส CChkBookDoc CCompDoc และคลาส CDefDoc สืบทอดมาจากคลาส CFixLenRecDoc ซึ่งคลาส CFixLenRecDoc ดำเนินการในการอ่าน เขียนข้อมูลส่วนหัวและระเบียนในแฟ้มข้อมูล รวมทั้งเลื่อนตัวชี้ไปยังระเบียนที่ต้องการ คลาส CChkBookDoc CCompDoc และ CDefDoc ดำเนิน

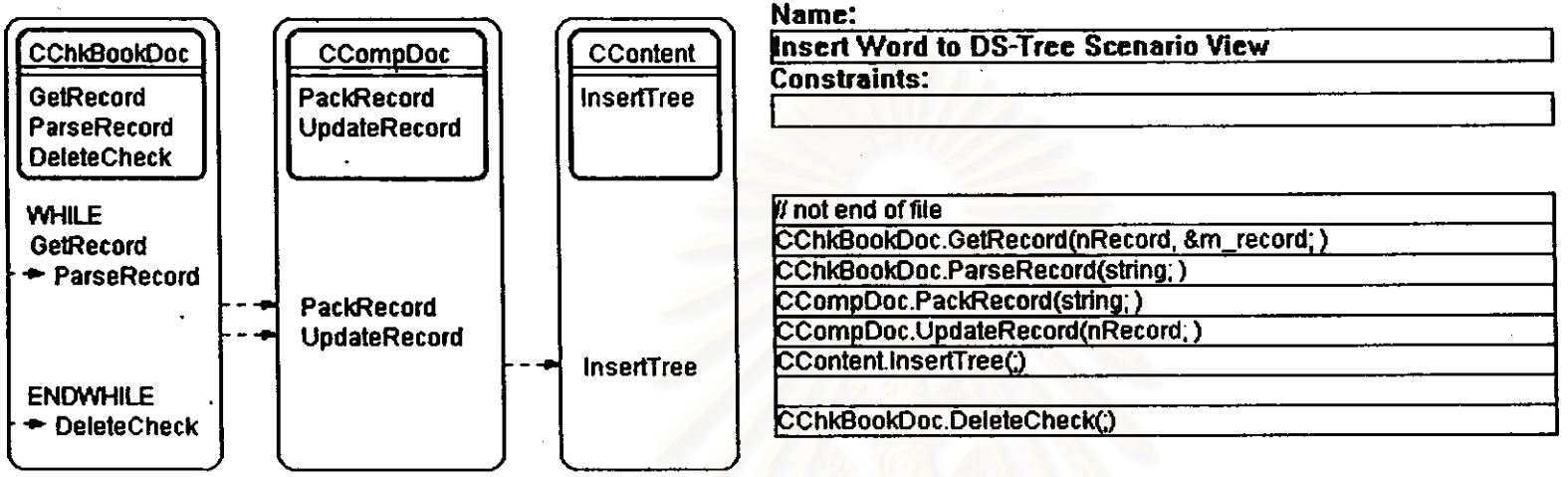
การในการนำข้อมูลจากสายอักขระบรรจุใน m_record โดยบริการ ParseRecord และนำข้อมูลจาก m_record บรรจุในสายอักขระโดยบริการ PackRecord

คลาส CDefDoc ดำเนินการกับแฟ้มข้อมูลที่ผู้ใช้กำหนดโครงสร้างในภาคแสดงความหมายเอง คลาส CChkBookDoc ดำเนินการในการรับข้อมูลที่ผู้ใช้ป้อนเข้ามาทีละระเบียนและจัดเก็บลงแฟ้มข้อมูล ส่วนคลาส CCompDoc จะรับข้อมูลจาก CChkBookDoc มาทีละระเบียนเมื่อมีการเพิ่มคำศัพท์ลงในดีเอส-ทรี จนกระทั่งหมดแฟ้มข้อมูล จากนั้นจะลบข้อมูลในแฟ้มข้อมูลเดิมทิ้งไป



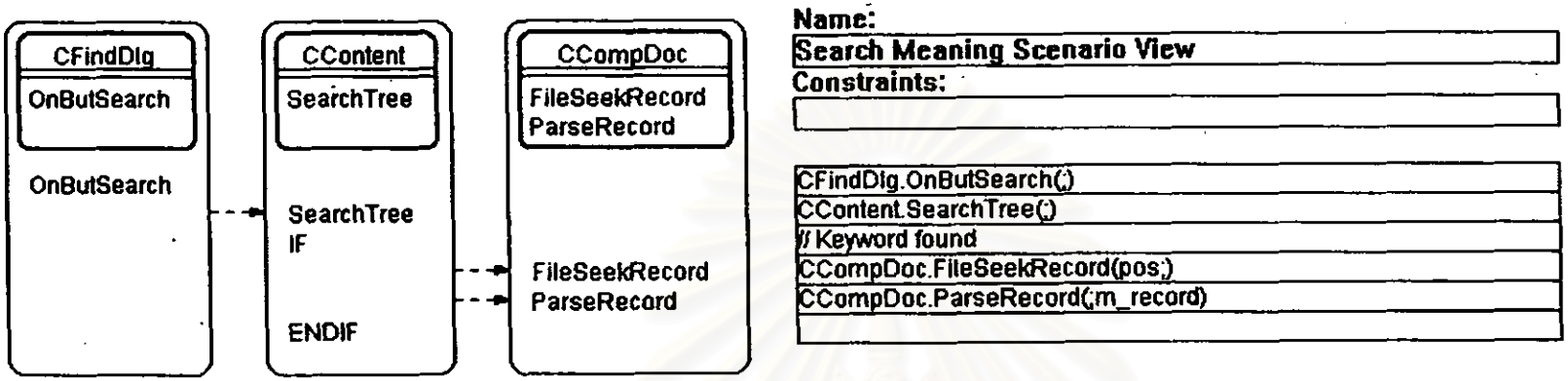
รูปที่ 6-6 การเชื่อมต่อพจนานุกรมกับจุฬาจารึก 78 (SI)

จากรูปที่ 6-6 คลาส CHook ดำเนินการในการเรียกจุฬาจารึก 78 ขึ้นมาทำงานโดยบริการ ExecCww ในขณะเดียวกันก็จะแทรกเมนูลงในจุฬาจารึกโดยบริการ InsertMenu เมื่อมีการเลือกเมนู พจนานุกรมก็ใช้บริการ SelectMenuDict เพื่อให้คลาส CHook ใช้บริการ OnSearch ทำให้คลาส CFindDlg แสดงจอภาพสำหรับสืบค้นคำศัพท์ ซึ่งบริการ OnButSerarch จะนำความหมายมาแสดง



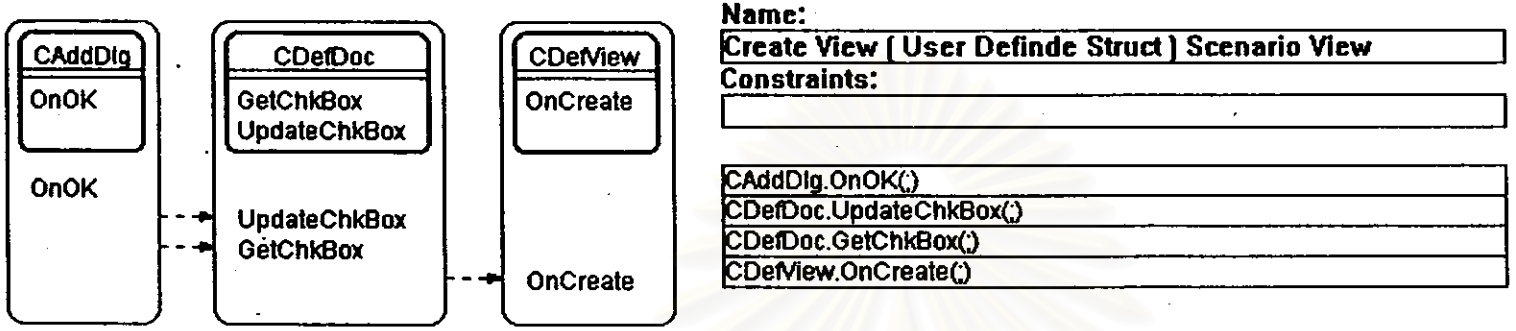
รูปที่ 6-7 แสดงการเพิ่มคำศัพท์ลงในดีเอส-ทรี

การนำคำศัพท์จากแฟ้มข้อมูลมาสร้างดีเอส-ทรี คลาส CChkBookDoc จะอ่านข้อมูลจากแฟ้มข้อมูลโดยบริการ GetRecord มาทีละระเบียน จากนั้นบริการ ParseRecord จะนำข้อมูลบรรจุลงในสวยอักขระ และคลาส CCompDoc ใช้บริการ PackRecord นำข้อมูลบรรจุลงใน m_reocrd เพื่อให้บริการ UpdateRecord บันทึกข้อมูลลงแฟ้มข้อมูลใหม่ ซึ่งในขณะเดียวกันนี้ คลาส Ccontent จะนำคำศัพท์ไปสร้างดีเอส-ทรี ไปด้วยโดยบริการ InsertTree เมื่อหมดแฟ้มข้อมูลคลาส CChkBookDoc จะลบข้อมูลในแฟ้มข้อมูลเดิมโดยบริการ DeleteCheck



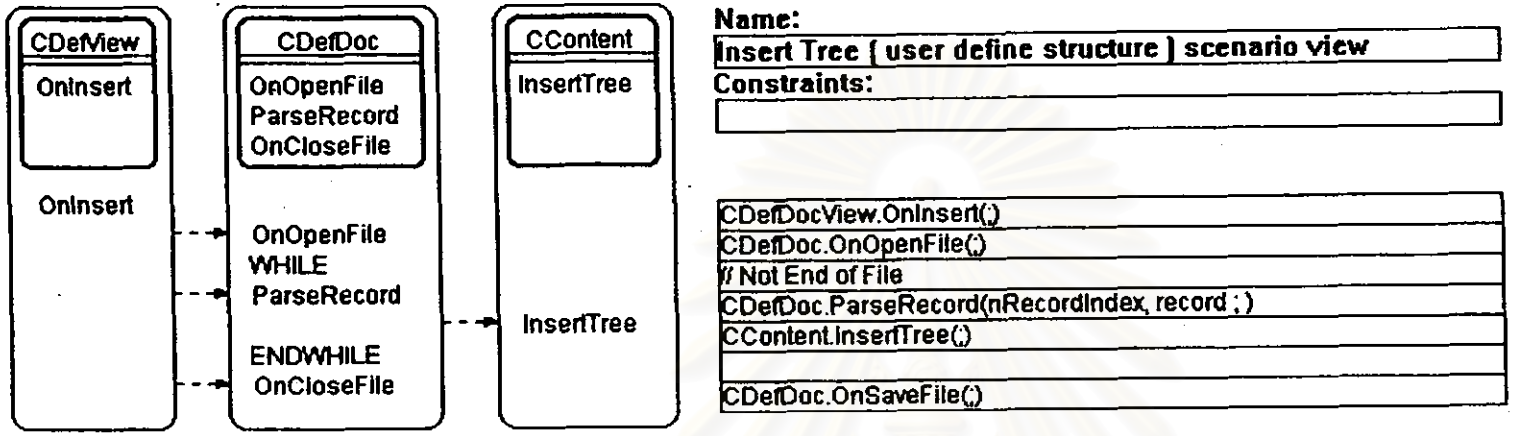
รูปที่ 6-8 แสดงการสืบค้นคำศัพท์และความหมาย

คลาส CFindDlg ดำเนินการในการสร้างจอภาพสำหรับสืบค้นคำศัพท์และความหมาย เมื่อผู้ใช้ป้อนคำศัพท์ที่ต้องการ บริการ OnButSearch จะร้องขอบริการจากคลาส CContent ให้สืบค้นคำศัพท์โดยใช้บริการ SearchTree ถ้าหากค้นพบคำศัพท์ คลาส CCompDoc จะใช้บริการ FileSeekRecord เพื่อชี้ไปยังระเบียนที่ต้องการในแฟ้มข้อมูลและนำข้อมูลระเบียนนั้นมาบรรจุในสายอักขระโดยบริการ ParseRecord เพื่อนำความหมายไปแสดงที่จอภาพต่อไป



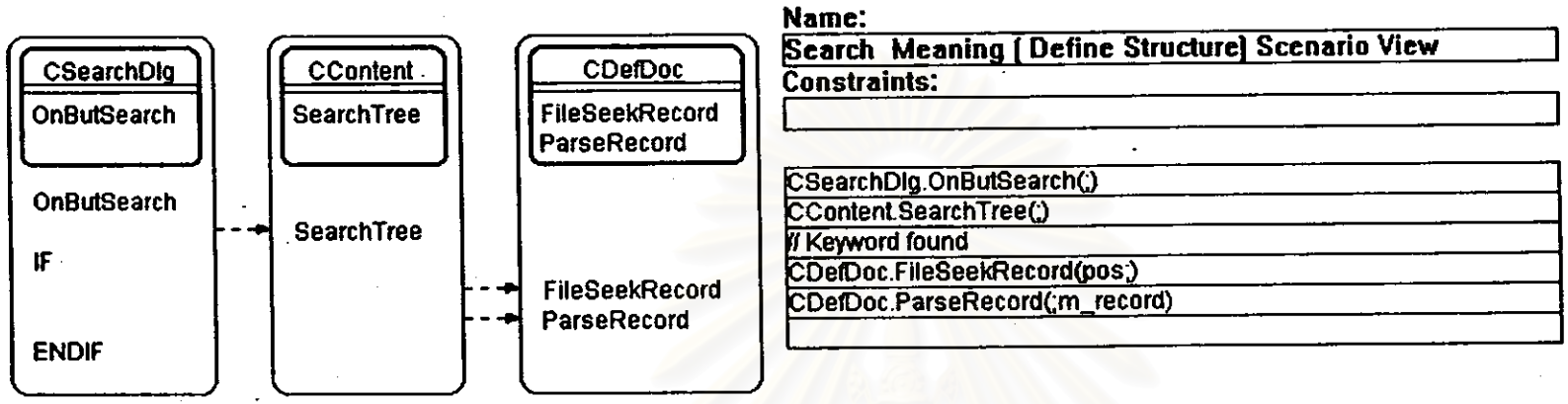
รูปที่ 6-9 การสร้างจอภาพสำหรับบันทึกคำศัพท์และความหมายกรณีผู้ใช้กำหนดโครงสร้างเอง

เมื่อผู้ใช้เลือกชนิดข้อมูลที่ต้องการให้มีในภาคแสดงความหมายแล้ว บริการ OnOK ของคลาส CAddDlg จะร้องขอบริการจากคลาส CDefDoc ให้บันทึกผลการเลือก โดยบริการ UpdateChkBox และเมื่อต้องการสร้างจอภาพสำหรับบันทึกข้อมูล บริการ GetChkBox ก็จะนำผลการเลือกชนิดข้อมูล มาสร้างช่องสำหรับบันทึกความหมายตามชนิดที่ได้เลือกไว้โดยบริการ OnCreate



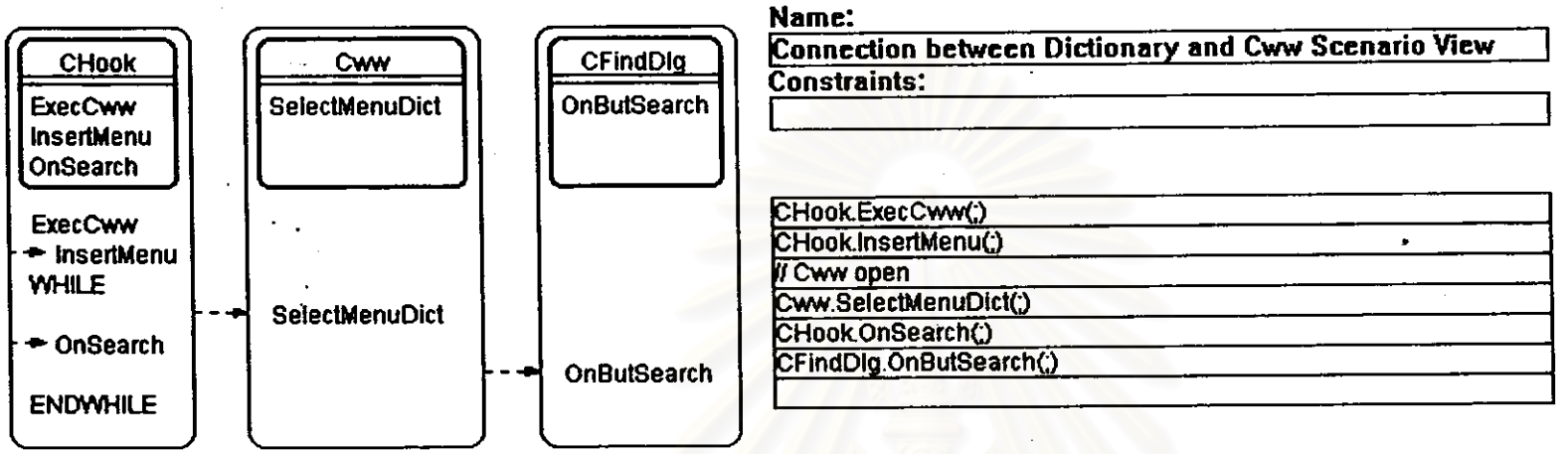
รูปที่ 6-10 แสดงการเพิ่มคำศัพท์ลงในดีเอส-ทรี กรณีผู้ใช้กำหนดโครงสร้างในภาคแสดงความหมายเอง

เมื่อต้องการเพิ่มคำศัพท์ลงในดีเอส-ทรี คลาส CDefView ใช้บริการ OnInsert เพื่อร้องขอบริการจากคลาส CDefDoc ใช้บริการ OnOpenFile เปิดแฟ้มข้อมูลและใช้บริการ ParseRecord นำข้อมูลจากแฟ้มข้อมูลมาทีละระเบียนเพื่อบรรจุลงในสายอักขระ และนำคำศัพท์ไปสร้างดีเอส-ทรี โดยบริการ InsertTree ของคลาส CContent จนกระทั่งหมดแฟ้มข้อมูลคลาส CDefDoc จะใช้บริการ OnCloseFile ปิดแฟ้มข้อมูล



รูปที่ 6-11 แสดงการสืบค้นคำศัพท์และความหมายกรณีผู้ใช้กำหนดโครงสร้างในภาคแสดงความหมายเอง

คลาส CSearchDlg ดำเนินการในการสร้างจอภาพสำหรับสืบค้นคำศัพท์และความหมาย เมื่อผู้ใช้ป้อนคำศัพท์ที่ต้องการ บริการ OnButSearch จะร้องขอบริการจากคลาส CContent ให้สืบค้นคำศัพท์โดยใช้บริการ SearchTree ถ้าหากค้นพบคำศัพท์ คลาส CDefDoc จะใช้บริการ FileSeekRecord เพื่อชี้ไปยังระเบียนที่ต้องการในแฟ้มข้อมูลและนำข้อมูลระเบียนนั้นมาบรรจุในสายอักขระโดยบริการ ParseRecord เพื่อนำความหมายไปแสดงที่จอภาพต่อไป



รูปที่ 6-12 แสดงการเชื่อมต่อระหว่างพจนานุกรมกับจุฬาจารึก

คลาส CHook จะเรียกจุฬาจารึก 78 ขึ้นมาทำงาน โดยบริการ ExecCww ในขณะเดียวกันก็จะแทรกเมนูพจนานุกรมลงในจุฬาจารึกไปด้วย คลาส CHook จะคอยตรวจสอบอยู่เสมอว่ามีการเลือกเมนูพจนานุกรมหรือไม่ ถ้าพบว่ามีการเลือกเมนูพจนานุกรม คลาส CHook จะใช้บริการ OnSearch ร้องขอบริการจากคลาส CFindDlg ให้บริการ OnButSeardh นำความหมายของคำศัพท์มาแสดงยังจอภาพ

การเชื่อมต่อระหว่างโปรแกรมพจนานุกรมอิเล็กทรอนิกส์กับจุฬาจารึก

เนื่องจากการพัฒนาโปรแกรมในส่วนนี้ไม่ได้เข้าไปแก้ไขโปรแกรมต้นฉบับของจุฬาจารึก แต่จะใช้วิธีแทรกเมนูลงไป และเมื่อเมนูพจนานุกรมอิเล็กทรอนิกส์ถูกเลือกโปรแกรมก็จะทำงาน ซึ่งผู้ใช้สามารถป้อนคำศัพท์หรือคัดลอกข้อความในจุฬาจารึกมาค้นหาความหมายในพจนานุกรมได้ นอกจากนี้ยังสามารถคัดลอกข้อความจากภาคแสดงความหมายไปแทรกข้อความในจุฬาจารึกได้ ซึ่งการทำงานของโปรแกรมมีขั้นตอน ดังนี้

1. เรียกโปรแกรมจุฬาจารึกขึ้นมาทำงาน
2. แทรกเมนูพจนานุกรมอิเล็กทรอนิกส์ลงในจุฬาจารึก
3. ตรวจสอบเมสเสจหากเกิดขึ้นเนื่องจากการเลือกเมนูพจนานุกรมอิเล็กทรอนิกส์ ก็จะเรียกโปรแกรมพจนานุกรมขึ้นมาทำงาน

การทำงานของโปรแกรมในส่วนนี้จะอยู่ใน Hooks.c และ Hooksdll.dll ซึ่งนำโปรแกรมตัวอย่างของ Visual C++ V.1.5 มาแก้ไข มีรายละเอียดการทำงาน ดังนี้

1. ประกาศฟังก์ชันกรองเมสเสจให้เป็นฟังก์ชันส่งออกในแฟ้มข้อมูล Hooksdll.def
2. รับค่าตำแหน่งแอดเดรสด้วยการเรียกใช้ฟังก์ชัน MakeProcInstance
3. เรียกฟังก์ชัน SetWindowsHookEx กำหนดชนิดของช่องเกี่ยวโยงเป็นชนิด WH_SYSMSGFILTER (ติดตั้งฟังก์ชันกรองเมสเสจทั้งระบบ) และตำแหน่งแอดเดรสของฟังก์ชัน(ได้จากฟังก์ชัน MakeProcInstance)
4. รับค่าที่ได้จากการเรียกใช้ฟังก์ชัน SetWindowsHookEx เก็บไว้ในตำแหน่งที่สำรองไว้ ค่าที่ได้นี้คือแฮนเดิลของฟังก์ชันกรองเมสเสจเดิม
5. เรียกโปรแกรมจุฬาจารึกขึ้นมาทำงานโดยใช้ฟังก์ชัน WinExec และแทรกเมนูพจนานุกรมอิเล็กทรอนิกส์ลงไปโดยใช้ฟังก์ชัน InsertMenu และกำหนดให้แทรกในตำแหน่งที่ 7
6. ตรวจสอบเมสเสจที่ส่งมาว่าเกิดขึ้นเนื่องจากการเลือกเมนูพจนานุกรมอิเล็กทรอนิกส์หรือไม่ ถ้าใช่จะเรียกโปรแกรมพจนานุกรมขึ้นมาทำงาน
7. หลังจากเลิกใช้งานจุฬาจารึกจะถอดฟังก์ชันกรองเมสเสจออกโดยใช้ฟังก์ชัน UnHookWindowsHookEx

เนื่องจากการที่จะทราบว่าผู้ใช้งานจุฬาจารึกมีการเลือกเมนูพจนานุกรมอิเล็กทรอนิกส์หรือไม่นั้น โปรแกรมของเราไม่สามารถที่จะรับเมสเสจจากจุฬาจารึกได้ เพราะเป็นเมสเสจจากต่างโปรแกรม ฉะนั้นจึงจำเป็นต้องใช้วิธีดักเมสเสจของทั้งระบบ โดยการนำช่องเกี่ยวโยงเข้ามาใช้ ซึ่งมีรายละเอียด ดังนี้

ช่องเกี่ยวโยง (Hook)

(จิรพัฒน์ จันทร์เจิดศักดิ์ และ วีระ นพนิราพาธ, 2521) ช่องเกี่ยวโยงเป็นกลไกการจัดการเมสเสจอย่างหนึ่งของวินโดว์ส ที่สามารถให้แอพพลิเคชั่นสามารถเข้าถึงเมสเสจที่ส่งกันไปมา วินโดว์สมีช่องเกี่ยวโยงหลายแบบแต่ละแบบมีความสามารถในการเข้าถึงเมสเสจแตกต่างกันไป การใช้ประโยชน์จากช่องเกี่ยวโยง คือ การติดตั้งฟังก์ชันกรองเมสเสจ (Filter Function) ซึ่งจะจัดการกับเมสเสจจากช่องเกี่ยวโยงก่อนที่เมสเสจนั้นจะถูกส่งไปยังฟังก์ชันประจำวินโดว์สที่อยู่ปลายทาง

ลูกโซ่ของฟังก์ชันกรองเมสเสจ

ลูกโซ่ของฟังก์ชันกรองเมสเสจเกิดจากการต่อฟังก์ชันกรองเมสเสจหลายๆ ฟังก์ชันเข้ากับช่องเกี่ยวโยงเดียวกันของระบบ โดยที่ตำแหน่งแรกสุดของลูกโซ่ก็คือช่องเกี่ยวโยงนั่นเอง และปลายของลูกโซ่ก็จะไปอยู่ที่ฟังก์ชันประจำวินโดว์ส เมื่อเราติดตั้งฟังก์ชันกรองเมสเสจ ฟังก์ชันดังกล่าวจะเกี่ยวเข้ากับช่องเกี่ยวโยงอันถือว่าเป็นจุดเริ่มต้นของลูกโซ่ รับเมสเสจแล้วส่งเมสเสจที่กรองแล้วโยงเข้ากับฟังก์ชันกรองเมสเสจที่เคยติดตั้งอยู่ (ซึ่งตอนนี้อยู่ในตำแหน่งถัดไป) ฟังก์ชันกรองถัดไปนั้นก็จะทำเช่นนี้ต่อกันไปเรื่อยๆ เป็นลูกโซ่ที่จะโยงเข้าฟังก์ชันประจำวินโดว์สในที่สุด

ฟังก์ชันกรองเมสเสจจะสามารถโยงตัวเองเข้ากับช่องเกี่ยวโยงด้วยฟังก์ชัน SetWindowsHookEx ในแต่ละครั้งที่เรียก ฟังก์ชันกรองเมสเสจเหล่านั้นก็จะถูกโยงไปยังจุดเริ่มต้นของลูกโซ่ (ซึ่งอาจจะมีฟังก์ชันกรองเมสเสจเกี่ยวอยู่) นั่นคือโยงต่อเข้ากับช่องเกี่ยวโยง และทุกครั้งที่แอพพลิเคชั่นติดตั้งฟังก์ชันกรองเมสเสจเข้ากับช่องเกี่ยว ก็จะต้องสำรองพื้นที่สำหรับเก็บตำแหน่งแอดเดรสของฟังก์ชันกรองเมสเสจที่เคยอยู่เดิม เพื่อส่งเมสเสจที่กรองแล้วโยงเข้ากับฟังก์ชันกรองที่อยู่ถัดไป

เมื่อฟังก์ชันกรองเมสเสจทำงานของตนเรียบร้อย ก็จะเรียกฟังก์ชัน DefHookProc ฟังก์ชันนี้จะใช้ค่าแอดเดรสที่เคยเก็บไว้เพื่อเรียกฟังก์ชันที่ควรจะอยู่ถัดไปในลูกโซ่

การถอดฟังก์ชันกรองเมสเสจออกจากลูกโซ่ แอพพลิเคชั่นเรียกใช้ฟังก์ชัน UnHookWindowsHookEx ด้วยชนิดของช่องเกี่ยวโยงและตัวชี้ไปยังฟังก์ชันกรองนั้น

รายละเอียดของแต่ละคลาส

1. คลาส CFixLenRecDoc

ลักษณะประจำ ประกอบด้วย

```
struct {
        UINT    nRecordCount;   ( จำนวนระเบียนในแฟ้มข้อมูล )
        UINT    nRecordLength;  ( ขนาดของระเบียน )
        UINT    nExtraHeaderLength;  ( ขนาดของ extraHeader )
} m_header
Cfile m_file ( แฟ้มข้อมูลจัดเก็บความหมาย )
```

บริการของวัตถุ ประกอบด้วย

virtual void WriteHeader(CFile* pFile, BOOL bNewHeader);

ทำหน้าที่บันทึกหรือแก้ไขข้อมูลใน m_header

virtual BOOL ReadHeader(CFile* pFile);

ทำหน้าที่อ่านข้อมูลจาก m_header

virtual UINT CreateNewRecord( );

ทำหน้าที่ในการสร้างระเบียนใหม่ เพิ่มค่า nRecordCount ใน m_header ปรับปรุงการแสดงช่องสำหรับบันทึกข้อมูลให้ว่างเพื่อเตรียมรับข้อมูลใหม่ และคืนค่าตัวชี้ไปยังระเบียนใหม่

UINT GetRecordCount( );

ทำหน้าที่อ่านจำนวนระเบียนภายในแฟ้มข้อมูลจาก m_header

virtual void GetRecord(UINT nRecordIndex, void* pRecord);

ทำหน้าที่อ่านข้อมูลระเบียนที่ nRecordIndex จากแฟ้มข้อมูล

virtual void UpdateRecord(CView* pSourceView, UINT nRecordIndex, void* pRecord);

ทำหน้าที่บันทึกข้อมูลระเบียนที่ nRecordIndex ลงแฟ้มข้อมูล

virtual void UpdateAllViewsWithRecord(CView* pSourceView, UINT nRecordIndex);

ทำหน้าที่ปรับปรุงการแสดงผลบนจอภาพด้วยข้อมูลระเบียนที่ nRecordIndex

virtual BOOL OnOpenDocument(const char* pszPathName);

ทำหน้าที่เปิดแฟ้มข้อมูลที่ระบุ ถ้าแฟ้มข้อมูลนั้นยังไม่มีจะสร้างใหม่

virtual void DeleteContents( );

ทำหน้าที่ในการปิดแฟ้มข้อมูล

virtual void FileSeekRecord(UINT nRecord);

ชี้ไปยังระเบียนที่ต้องการ

2. คลาส CChkBookDoc

ลักษณะประจำ ประกอบด้วย

```
UINT m_nActiveRecord;    ( หมายเลขระเบียนที่แอคทีฟอยู่ )
struct
{
        DWORD dwFileSignature; ( ใช้ตรวจสอบว่าเป็นแฟ้มข้อมูลที่ต้องการ )
        UINT nFirstCheckNo;  ( ดรรชนีของระเบียนแรก )
} m_extraHeader;
struct
{
        char    szWord ( คำศัพท์ )
        char    szMean ( ความหมาย )
        char    szSam  ( ตัวอย่างวิธีการใช้งาน )
        char    szSyno  ( คำเหมือน )
        char    szAnto  ( คำตรงกันข้าม )
} m_record;
 CFile fcomp;  ( แฟ้มข้อมูลจัดเก็บความหมาย )
```

บริการของวัตถุ ประกอบด้วย

BOOL OnOpenDocument(const char* pszPathName)

นำฟังก์ชัน OnOpenDocument จาก CFixLenRecDoc มาเพิ่มการบันทึกเส้นทางของแฟ้มข้อมูล ในแฟ้มข้อมูล (. INI)

BOOL OnSaveDocument(const char* pszPathName)

ในขณะที่จัดเก็บแฟ้มข้อมูล จะบันทึกเส้นทางของแฟ้มข้อมูล ในแฟ้มข้อมูล (. INI)

BOOL SaveModified( )

เมื่อมีการแก้ไขข้อมูลและยังไม่ได้จัดเก็บข้อมูล จะสอบถามยืนยันการจัดเก็บข้อมูล

void UpdateIniFileWithDocPath(const char* pszPathName)

ปรับปรุงเส้นทางของแฟ้มข้อมูล ในแฟ้มข้อมูล (INI)

void* OnCreateNewRecord(int nNewRecordIndex)

เตรียมเนื้อที่ระเบียนใหม่ในหน่วยความจำหลัก

BOOL OnReadExtraHeader( )

ทำหน้าที่อ่านข้อมูล m_extraHeader จากแฟ้มข้อมูล

void OnWriteExtraHeader(BOOL bNewHeader)

ทำหน้าที่บันทึกข้อมูลลงใน m_extraHeader

void GetCheck(UINT nCheckNo, CString strWord, CString strMean, CString strSam, CString strSyno, CSring strAnto)

ทำหน้าที่นำข้อมูล m_record มาแสดงที่จอภาพ

void UpdateCheck(CView* pSourceView, UINT nCheckNo,
const char* szWord, const char* szMean, const char* szSam,
const char* szSyno, const char* szAnto)

ทำหน้าที่บันทึกข้อมูลระเบียนที่ nCheckNo จากจอภาพลงในแฟ้มข้อมูล

void ChangeSelectionNextCheckNo(BOOL bNext)

ทำหน้าที่เลื่อนการแสดงผลของจอภาพไปยังระเบียนถัดไป ถ้าหากยังไม่ได้จัดเก็บข้อมูลระเบียนปัจจุบันจะสอบถามยืนยันการจัดเก็บ

void ChangeSelectionToCheckNo(UINT nNewActiveCheckNo)

ทำหน้าที่แสดงข้อมูลบนจอภาพด้วยข้อมูลระเบียนที่ระบุ

BOOL MaybeCommitDirtyCheck( )

ทำหน้าที่สอบถามยืนยันการจัดเก็บข้อมูล

void PackRecord(const char* szWord, const char* szMean,
const char* szSam, const char* szSyno, const char* szAnto)

ทำหน้าที่บันทึกข้อมูล szWord, szMean, szSam, szSyno และ szAnto ลงใน m_reocrd

void ParseRecord(CString& strWord, CString& strMean,
CString& strSam, CString& strSyno, CString& strAnto)

ทำหน้าที่นำข้อมูลใน m_record บันทึกลงใน strWord, strMean, strSam, strSyno และ strAnto

UINT CheckNoToRecordIndex(UINT nCheckNo)

ทำหน้าที่เปลี่ยนค่าจาก nCheckNo เป็น nRecordIndex

UINT RecordIndexToCheckNo(UINT nRecordIndex)

ทำหน้าที่เปลี่ยนค่าจาก nRecordIndex เป็น nCheckNo

UINT GetActiveCheckNo( )

คืนค่าดรรชนีของระเบียนปัจจุบัน

UINT GetFirstCheckNo( )

คืนค่าดรรชนีของระเบียนแรก

UINT GetLastCheckNo( )

คืนค่าตรรชนีของระเบียนสุดท้าย

void NewCheck( )

ทำหน้าที่สร้างระเบียนใหม่ แต่จะตรวจสอบก่อนถ้าหากยังไม่ได้จัดเก็บข้อมูลระเบียนปัจจุบัน จะสอบถามยืนยันการจัดเก็บข้อมูลปัจจุบันก่อน

void OnNextCheck( )

ทำให้จอภาพแสดงข้อมูลระเบียนถัดไป

void OnUpdateNextCheck(CCmdUI* pCmdUI)

ทำให้เมนู Next Content ใช้งานได้

void OnPrevCheck( )

ทำให้จอภาพแสดงข้อมูลระเบียนก่อนหน้า

void OnUpdatePrevCheck(CCmdUI* pCmdUI)

ทำให้เมนู Prev Content ใช้งานได้

3. คลาส CBookView

บริการของวัตถุ ประกอบด้วย

void OnUpdate(CView* pSender, LPARAM lHint = 0L, CObject* pHint = NULL);

ทำหน้าที่ปรับปรุงการแสดงผลข้อมูลบนจอภาพ ทุกครั้งที่ข้อมูลเปลี่ยนแปลง

void GetRowWidthHeight(CDC* pDC, int& nRowWidth, int& nRowHeight);

คืนค่าความกว้างและความสูงของแถว

int GetActiveRow( );

คืนค่าข้อมูลในแถวที่ทำงานอยู่

int GetRowCount( );

หาจำนวนแถวทั้งหมดของข้อมูล

void OnDrawRow(CDC* pDC, int nRowNo, int'y, BOOL bSelected);

แสดงข้อมูลของระเบียนทั้งหมดบนจอภาพและกำหนดสีที่เป็นแถบสว่างสำหรับระเบียนปัจจุบัน

void ChangeSelectionNextRow(BOOL bNext);

เลื่อนแถบสว่างไปยังแถวถัดไป

void ChangeSelectionToRow(int nRow);

เลื่อนแถบสว่างไปยังแถวที่ต้องการ

4. คลาส CCheckView

ลักษณะประจำ ประกอบด้วย

UINT m_nCheckNo; (หมายเลขระเบียน)

CString m_strWord; (คำศัพท์)

CString m_strMean; (ความหมาย)

CString m_strSam; (ตัวอย่างวิธีการใช้งาน)

CString m_strSyno; (คำเหมือน)

CString m_strAnto; (คำตรงข้าม)

บริการของวัตถุ ประกอบด้วย

void OnUpdate(CView* pSender, LPARAM lHint, CObject* pHint);

ทำหน้าที่ปรับปรุงข้อมูลในจอภาพทุกครั้งที่ข้อมูลเปลี่ยนแปลง

BOOL MaybeCommitDirtyCheck( );

ทำหน้าที่ในการตรวจสอบว่ามีการจัดเก็บข้อมูลหรือยัง ถ้ายังไม่ได้ยืนยันการจัดเก็บข้อมูล จะสอบถามยืนยันการจัดเก็บข้อมูล หลังจากยืนยันแล้วจะปรับปรุงข้อมูลในจอภาพใหม่

5. คลาส CCompDoc

ลักษณะประจำ ประกอบด้วย

```
UINT    m_nActiveRecord;  ( หมายเลขระเบียนปัจจุบัน )
struct
{
        UINT nRecordCount;   ( จำนวนระเบียนในแฟ้มข้อมูล )
        UINT nRecordLength;  ( ความยาวของระเบียน )
        UINT nExtraHeaderLength;   ( ความยาวของ ExtraHeader )
} m_header;
struct
{
        DWORD dwFileSignature;  ( ใช้ตรวจสอบว่าเป็นแฟ้มข้อมูลที่ต้องการ )
        UINT nFirstCheckNo;     ( ดรรชนีของระเบียนแรก )
} m_extraHeader;
struct
{
        char    szWord[30];     ( คำศัพท์ )
        char    szMean[200];    ( ความหมาย )
        char    szSam[200];     ( ตัวอย่างวิธีการใช้งาน )
        char    szSyno[30];     ( คำเหมือน )
        char    szAnto[30];     ( คำตรงข้าม )
} m_record;
CFile m_file;     ( แฟ้มข้อมูลจัดเก็บความหมาย )
```

บริการของวัตถุ ประกอบด้วย

```
void*   OnCreateNewRecord( int nNewRecordIndex );
```

ทำหน้าที่เตรียมเนื้อที่สำหรับ ระเบียนใหม่ในหน่วยความจำหลัก

```
void    WriteHeader( );
```

ทำหน้าที่บันทึกข้อมูลใน m_header

BOOL OnReadExtraHeader( );
ทำหน้าที่อ่านข้อมูลจาก m_extraHeader

void OnWriteExtraHeader( );
ทำหน้าที่บันทึกข้อมูลลงใน m_extraHeader

void UpdateRecord(UINT nRecordIndex);
ทำหน้าที่บันทึกข้อมูลระเบียนที่ระบุลงแฟ้มข้อมูล

void FileSeekRecord(UINT nRecord);
ไปยังระเบียนที่ระบุ

BOOL OnOpenFile( );
ทำหน้าที่เปิดแฟ้มข้อมูล

void OnSaveFile( );
ทำหน้าที่ปิดแฟ้มข้อมูล

void PackRecord(const char* szWord, const char* szMean, const char* szSam,
const char* szSyno, const char* szAnto);
ทำหน้าที่บันทึกข้อมูล szWord, szMean, szSam, szSyno และ szAnto ลงใน m_record

void ParseRecord(CString& strWord, CString& strMean,
CString& strSam, CString& strSyno, CString& strAnto);
ทำหน้าที่นำข้อมูลใน m_record บันทึกลงใน strWord, strMean, strSam, strSyno และ strAnto

6. คลาส CContent
ลักษณะประจำ ประกอบด้วย

long int* doub_base; (อะเรย์ของเบส)
int * doub_check; (อะเรย์ของเช็ค)
int da_size; (ขนาดสูงสุดของอะเรย์เช็ค)

long int POS; (ตำแหน่งของตัวอักษรในคำศัพท์)

UINT * doub_value; (อะเรย์เก็บดรรชนีชี้ไปยังระเบียนที่จัดเก็บความหมาย)

CFile ftail (แฟ้มข้อมูลจัดเก็บชิงเกิลสตริง)

CFile fbase (แฟ้มข้อมูลจัดเก็บอะเรย์เบส)

CFile fcheck (แฟ้มข้อมูลจัดเก็บอะเรย์เช็ค)

CFile fvalue (แฟ้มข้อมูลจัดเก็บดรรชนีของระเบียน)

บริการของวัตถุ ประกอบด้วย

int wordtoint(CString bword, int position);

จะส่งคืนค่าตัวเลขของตัวอักษรใน bword ณ ตำแหน่ง position

bword แทนคำศัพท์หรือส่วนหนึ่งของคำศัพท์

position แทนตำแหน่งของตัวอักษรใน bword

int ctoint(char x);

จะส่งคืนค่าตัวเลขของตัวอักษร x

int Rd_Base(int base_pos);

อ่านค่าจาก doub_base ณ ตำแหน่ง base_pos

int Rd_Check(int chk_pos);

อ่านค่าจาก doub_check ณ ตำแหน่ง chk_pos

UINT Rd_Value(int val_pos);

อ่านค่าจาก doub_value ณ ตำแหน่ง val_pos

int A_Insert(int a_r, CString a_str, UINT index);

ทำหน้าที่เพิ่มคำศัพท์หลังจากที่สืบค้นคำศัพท์แล้วไม่พบ เนื่องจากการสร้างมัลติโหนดไม่สำเร็จ

a_r แทนหมายเลขโหนดปัจจุบัน

a_str แทนคำศัพท์ส่วนที่เหลือจากการสร้างมัลติโหนด

index ดรรชนีชี้ไปยังระเบียนของแฟ้มข้อมูล

ขั้นตอนการเพิ่มข้อมูลของฟังก์ชัน A_Insert (a_r, a_str, index)

1. กำหนดค่าของหมายเลขโหนดถัดไป

   t = BASE[r] + a_h

2. ตรวจสอบค่าของ CHECK[t] = 0 ทำงานในข้อ 4

   ถ้าค่าของ CHECK[t] ไม่เท่ากับ 0 ทำงานในข้อ 3

3. เปลี่ยนค่าของ BASE[r] และ BASE[t] ซึ่ง k = CHECK[t] ตามขั้นตอนต่อไปนี้

   3.1 ใช้ฟังก์ชัน SetList(r) เพื่อหาค่า Rlist และ ใช้ฟังก์ชันSetList(k) เพื่อหาค่า Klist

   3.2 ตรวจสอบจำนวนสมาชิกของ Rlist + 1 ว่ามากกว่า จำนวนสมาชิกของ Klist หรือไม่

   ถ้าเงื่อนไขเป็นจริง ทำงานในข้อ 3.3

   ถ้าเงื่อนไขเป็นเท็จ ทำงานในข้อ 3.4

   3.3 ใช้ฟังก์ชัน Modify(r, r, {a_h} , Rlist,' index) เพื่อหาค่า BASE[k] โดยที่ต้องอยู่ภายใต้เงื่อนไข CHECK[BASE[r]+b] = 0 และ b เป็นสมาชิกใน Rlist U {a_h} ต่อไปทำงานในข้อ 4

   3.4 ใช้ฟังก์ชัน Modify(r, k, , Klist, index) เพื่อหาค่า BAS[k] ซึ่งอยู่ภายใต้เงื่อนไข CHECK[BASE[r] + a_h] ไม่เท่ากับ CHECK[BASE[k] + b] และ b เป็นสมาชิกใน Klist ต่อไปทำงานในข้อ 4

4. ใช้ฟังก์ชัน Ins_Str(r, $a_h a_{h+1}$ .....a_{n+1}, POS, index) เพื่อสร้างโหนดในดับเบิลอะเรย์ตามสมการ $g(s_r, a_h)$ และจัดเก็บ a_{h+1}.....a_{n+1} ในอะเรย์เทล

void Set_List(int r, int sym_list[Maxchar]);

ให้ค่าเซทของ a ที่สอดคล้องกับสมการ CHECK[BASE[r] + a] = r

int Insert_Str(int n_h, CString e_1e_2....e_q, int d_pos, UINT w_value);

เป็นฟังก์ชันที่ใช้เก็บชิงเกิลสตริงในอะเรย์เทล ณ ตำแหน่งที่ระบุ (d_pos) และหากตำแหน่งนั้นมีข้อมูลอยู่แล้ว จะส่งตำแหน่งสุดท้ายของอะเรย์เทลตำแหน่งใหม่กลับมา ขั้นตอนการทำงานมีดังนี้

1. กำหนดค่าโหนดถัดไป

   t = BASE[n_h] + e_1

2. กำหนดค่า BASE[t] = -d_pos และ CHECK[t] = n_h

3. ใช้ฟังก์ชัน Str_Tail(d_pos, e_2e_3 .....e_q) เพื่อเก็บสายอักขระ e_2e_3 .....e_q ใน

อะเรย์เทล ซึ่งฟังก์ชันดังกล่าวจะคืนค่า POS ซึ่งเป็นดรรชนีสูงสุดของอะเรย์เทล หลังจากที่เพิ่มสตริงแล้วกลับมา

long int Str_Tail(int p, CString y);

เป็นฟังก์ชันที่ใช้สำหรับจัดเก็บสายอักขระ y ในตำแหน่ง p ของอะเรย์เทล เมื่อจัดเก็บสายอักขระแล้วจะให้ค่า ดังนี้

ถ้า p = POS จะส่งค่า POS + ความยาวของสายอักขระ y กลับ

ถ้า p < POS จะส่งค่า POS ค่าเดิมกลับ

int Modify(int current_s, unsigned int m_h, int add[Maxchar], int org_list [Maxchar], UINT w_value);

เป็นฟังก์ชันที่ใช้สำหรับการหาค่าดรรชนีของอะเรย์เบสค่าใหม่ เนื่องจากมีการใช้อะเรย์เบสซ้ำ และทำการปรับค่าของอะเรย์เบสและอะเรย์เช็คใหม่ ซึ่งมีขั้นตอนดังนี้

1. กำหนดค่า old_base = BASE[m_h]
   หาค่า BASE[m_h] ใหม่ โดยใช้ฟังก์ชัน X_Check(ADD U ORG)
2. ทำงานซ้ำตั้งแต่ข้อ 3 ถึงข้อ 6 จนกระทั่งค่า c ใน ORG หมด
3. กำหนดค่า t = old_base + c
   กำหนดค่า t' = BASE[m_h] + c
   กำหนดค่า BASE[t'] = BASE[t] และ CHECK[t'] = m_h
   $g(s_h, c) = s_t$ ถูกกำหนดใหม่เป็น $g(s_h, c) = s_{t'}$
4. ตรวจสอบว่า BASE[t] มากกว่า 0 หรือไม่
   ถ้าเงื่อนไขเป็นจริง ทำงานในข้อ 5
   ถ้าเงื่อนไขเป็นเท็จ ทำงานในข้อ 6
5. กำหนดค่าของ CHECK[q] = t'
   โดยที่ q เป็นไปตามเงื่อนไข CHECK[BASE[t] + b] = t และ q = BASE[t] + b
   และกำหนดค่าของ t' = current_s (ถ้า t = current_s)
6. ให้ค่า BASE[t] = 0 และ CHECK[t] = 0
7. ออกจากฟังก์ชันและส่งค่า Current_s กลับ

int B_Insert(int b_r, CString sep_str, CString rem_str, CString b_str,UINT index);

มีหน้าที่ในการเพิ่มคำศัพท์ใหม่ลงใน ดี-เอส ทรี หลังจากสืบค้นคำศัพท์แล้วไม่พบ เนื่องจากเปรียบเทียบซิงเกิลสตริงกับคำศัพท์ส่วนที่เหลือแล้วไม่ตรงกัน

b_r	แทนโหนดปัจจุบัน
sep_str	แทนจำนวนตัวอักษรตั้งแต่ส่วนต้นที่เหมือนกันของ rem_str และ b_str
rem_str	แทนคำศัพท์ส่วนที่เหลือ
b_str	แทนซิงเกิลสตริง
index	แทนดรรชนีชี้ไปยังระเบียนที่เก็บความหมาย

ขั้นตอนการทำงานของฟังก์ชัน B_Insert (int b_r, CString sep_str, CString rem_str, CString b_str,UINT index)

1. กำหนดค่า old_pos = -BASE[b_r]
2. สร้างเส้นทางเดินของโหนดโดยใช้ตัวอักษรจากสายอักขระที่อ่านเข้ามา ตามขั้นตอนต่อไปนี้
   2.1 หาค่า BASE[b_r] จากฟังก์ชัน X_Check(a_{h+i}) โดยที่ 1<= i <= k
   2.2 กำหนดค่า CHECK[BASE[b_r] + a_{h+i}] = b_r
   2.3 คำนวณค่าโหนด b_r ใหม่จาก BASE[b_r] + a_{h+i} ให้สร้างเส้นทางเดินของโหนดจนกระทั่ง i = k
3. หาค่า BASE[b_r] ใหม่จากฟังก์ชัน X_Check(a_{h+k+1} , b_1)
4. นำสายอักขระส่วนที่ได้จากอะเรย์เทลเก็บในอะเรย์เทลที่ตำแหน่งเดิม โดยใช้ฟังก์ชัน Ins_Str(b_r, b_2 .....b_m , Old_Pos, index)
5. เก็บคำศัพท์ส่วนที่เหลือจากการสร้างมัลติโหนดและซิงเกิลสตริงในอะเรย์เทล ที่ตำแหน่งใหม่โดยใช้ฟังก์ชัน Ins_Str(b_r, a_{h+k+2} .....a_n a_{n+1}, POS, index)

   ในตอนเริ่มต้นทั้งอะเรย์เบสและอะเรย์เช็คจะมีเพียงรายการเดียวเท่านั้น คือ BASE[1] = 1 และ CHECK[1] = 1 เมื่อมีการเพิ่มคำศัพท์เข้าไปโครงสร้างของข้อมูลจะเพิ่มขนาดขึ้นเรื่อยๆ

void Init_Rtn( );

จองพื้นที่และกำหนดค่าเริ่มต้นให้ doub_base, doub_check, doub_value

int Push_Base(int base_pos, long int base_val, UINT w_value);

บันทึกค่าใน doub_base ด้วยค่า base_val ณ ตำแหน่ง base_pos

และบันทึกค่าใน doub_value ด้วยค่า w_value ณ ตำแหน่ง base_pos ซึ่งค่า w_value นี้
คือ ดรรชนีซึ่งชี้ไปยังระเบียนที่เก็บความหมาย

int Push_Check(int chk_pos, int chk_val);
บันทึกค่าใน doub_check ด้วยค่า chk_val ณ ตำแหน่ง chk_pos

int Rd_Array( );
อ่านค่าจากแฟ้มข้อมูล BASE.DAT CHECK.DAT VALUE.DAT และบันทึกลงใน doub_base,
doub_check และ doub_value ตามลำดับ

void Wr_Array( );
บันทึกค่าจาก doub_base, doub_check ,doub_value ลงในแฟ้มข้อมูล BASE.DAT, CHECK.DAT
และ VALUE.DAT ตามลำดับ

CString Fetch_Str(int post);
มีหน้าที่ในการดึงซิงเกิลสตริงมาจากอะเรย์เทล เริ่มจากตำแหน่ง post จนถึง
ตำแหน่ง post + k โดยที่
post คือ ค่าสัมบูรณ์ของอะเรย์เบสในกรณีที่ค่า BASE[b_r] < 0
k คือ ตำแหน่งสุดท้ายของซิงเกิลสตริง k >= 0
TAIL[post + k] = #

int Str_Cmp(CString str_1, CString str_2);
มีหน้าที่ในการเปรียบ str_1 กับ str_2 ซึ่งจะให้ผล ดังนี้
คืนค่า -1 ถ้า str_1 เหมือนกับ str_2
และคืนค่า n (n คือจำนวนตัวอักษรสูงสุดที่ str_1 เหมือนกับ str_2)

7. คลาส CDefDoc
ลักษณะประจำ ประกอบด้วย
int* def_array;
UINT m_nActiveRecord;
CFile m_file, deffile;

```
- struct
        {
                UINT nRecordCount;        ( จำนวนระเบียนในแฟ้มข้อมูล )
                UINT nRecordLength;       ( ความยาวของระเบียน )
                UINT nExtraHeaderLength;  ( ความยาวของ ExtraHeader )
} m_header;
struct entry {
                char* m_word;   ( พอยน์เตอร์ชี้ไปยังตัวแปร m_word )
                char* m_type;   ( พอยน์เตอร์ชี้ไปยังตัวแปร m_type )
                char* m_pron;   ( พอยน์เตอร์ชี้ไปยังตัวแปร m_pron )
                char* m_def;    ( พอยน์เตอร์ชี้ไปยังตัวแปร m_def )
                char* m_syno;   ( พอยน์เตอร์ชี้ไปยังตัวแปร m_syno )
                char* m_anto;   ( พอยน์เตอร์ชี้ไปยังตัวแปร m_anto )
                char* m_abb;    ( พอยน์เตอร์ชี้ไปยังตัวแปร m_abb )
                char* m_sam;    ( พอยน์เตอร์ชี้ไปยังตัวแปร m_sam )
                char* m_date;   ( พอยน์เตอร์ชี้ไปยังตัวแปร m_date )
};
entry* tab;
int maxtype;
int maxpron;
int maxdef;
int maxsyno;
int maxanto;
int maxabb;
int maxsam;
int maxdate;
struct
        {
                DWORD dwFileSignature;  ( ใช้ตรวจสอบว่าเป็นแฟ้มข้อมูลที่ต้องการ )
                UINT nFirstCheckNo;     ( ตรรชนีของระเบียนแรก )
        } m_extraHeader;
```

บริการของวัตถุ ประกอบด้วย

BOOL OnReadExtraHeader( ):

ทำหน้าที่อ่านข้อมูล m_ExtraHeader จากแฟ้มข้อมูล เพื่อตรวจสอบว่าเป็นประเภทของแฟ้มข้อมูลถูกต้อง

void OnWriteExtraHeader( );

ทำหน้าที่บันทึกข้อมูล m_ExtraHeader ลงในแฟ้มข้อมูล

void UpdateRecord(UINT nRecordIndex);

ทำหน้าที่บันทึกข้อมูลระเบียนที่ nRecordIndex ลงในแฟ้มข้อมูล

void OnNewRecord( );

ทำหน้าที่จองพื้นที่ระเบียนใหม่ ในแฟ้มข้อมูล

BOOL MaybeCommitDirtyCheck( );

ทำหน้าที่สอบถามยืนยันการจัดเก็บข้อมูล

void PackRecord(UINT nRecord, const char* szWord, const char* szType, const char* szPron, const char* szDef, const char* szSyno, const char* szAnto, const char* szAbb, const char* szSam, const char* szDate);

ทำหน้าที่บันทึกข้อมูล szWord, szType, szPron, szDef, szSyno, szAnto, szAbb, szSam และ szDate ลงใน tab

void ParseRecord(UINT nRecordIndex, CString& m_editword, CString& m_edittype, CString& m_editpron, CString& m_editdef, CString& m_editsyno, CString& m_editanto, CString& m_editabb, CString& m_editsam, CString& m_editdate);

ทำหน้าที่นำข้อมูลจากแฟ้มข้อมูล ณ ระเบียนที่ nRecordIndex บันทึกลงใน m_editword, m_edittype, m_editpron, m_editdef, m_editsyno, m_editanto, m_editabb, m_editsam และ m_editdate

void UpdateChkBox(int dmaxtype, int dmaxpron, int dmaxdef, int dmaxsyno,
int dmaxanto, int dmaxabb, int dmaxsam, int dmaxdate);

ทำหน้าที่บันทึกข้อมูลการเลือกชนิดของข้อมูลแสดงความหมาย

void GetChkBox( );

ทำหน้าที่อ่านข้อมูลการเลือกชนิดของข้อมูลแสดงความหมาย

UINT GetActiveCheckNo( );

ทำหน้าที่หาระเบียนปัจจุบัน

UINT CheckNoToRecordIndex(UINT nCheckNo);

ทำหน้าที่เปลี่ยน nCheckNo เป็น nRecordIndex

UINT RecordIndexToCheckNo(UINT nRecordIndex);

ทำหน้าที่เปลี่ยน nRecordIndex เป็น nCheckNo

void UpdateCheck(CView* pSourceView, UINT nCheckNo, const char* szWord,
const char* szType, const char* szPron, const char* szDef,const char* szSyno,
const char* szAnto, const char* szAbb, const char* szSam, const char* szDate);

ทำหน้าที่บันทึกข้อมูล szWord, szType, szPron, szDef, szSyno, szAnto, szAbb, szSam และ szDate ลงใน tab โดยเรียกใช้บริการ PackRecord

void OpenDefFile( );

ทำหน้าที่เปิดแฟ้มข้อมูล

void SaveDefFile( );

ทำหน้าที่ปิดแฟ้มข้อมูล

void InsertDef( );

ทำหน้าที่เพิ่มคำศัพท์ลงใน ดีเอส–ทรี

UINT GetLastCheckNo( );

ทำหน้าที่หาหมายเลขระเบียนสุดท้าย

UINT GetFirstCheckNo( );

ทำหน้าที่หาหมายเลขระเบียนแรก

void CreateDefFile( );

ทำหน้าที่สร้างแฟ้มข้อมูลใหม่

void ChangeSelectionToCheckNo(UINT nNewActiveCheckNo);

ทำหน้าที่เลื่อนการแสดงผลของจอภาพไปยังระเบียนที่ระบุ โดยที่จะตรวจสอบก่อนว่าระเบียนปัจจุบันได้รับการบันทึกหรือยัง ถ้ายังจะสอบถามยืนยันการบันทึกข้อมูล

UINT RecordCount( );

ทำหน้าที่หาจำนวนระเบียนในแฟ้มข้อมูล

void PrevCheckNo( );

ทำหน้าที่หาดรรชนีของระเบียนก่อนหน้า

void NextCheckNo( );

ทำหน้าที่หาดรรชนีของระเบียนถัดไป

8. คลาส CDefView

ลักษณะประจำ ประกอบด้วย

int maxtype;

int maxpron;

int maxdef;

int maxsyno;

int maxanto;

int maxabb;

int maxsam;

int maxdate;

UINT m_nCheckNo;

CEdit m_editword;

CEdit m_edittype, m_editpron;

CEdit m_editdef, m_editsyno, m_editanto, m_editabb, m_editsam, m_editdate;

CStatic m_stword, m_sttype, m_stpron;

CStatic m_stdef, m_stsyno, m_stanto, m_stabb, m_stsam, m_stdate;

UINT m_nActiveRecord;

บริการของวัตถุ ประกอบด้วย

void OnUpdate(CView*, LPARAM lHint, CObject* pHint);

ทำหน้าที่ปรับปรุงการแสดงผลของจอภาพทุกครั้งที่ ข้อมูลในดอกคิวเมนท์เปลี่ยนแปลง

BOOL MaybeCommitDirtyCheck( );

ทำหน้าที่ตรวจสอบการจัดเก็บข้อมูลระเบียนปัจจุบัน หากยังไม่ได้ทำการจัดเก็บจะสอบถาม ยืนยันการจัดเก็บข้อมูล และหากผู้ใช้งานยืนยันการจัดเก็บข้อมูล จะทำการบันทึกข้อมูลระเบียนใหม่ลงในแฟ้มข้อมูล

int OnCreate(LPCREATESTRUCT lpCreateStruct);

ทำหน้าที่แสดงช่องสำหรับบันทึกชนิดข้อมูลแสดงความหมาย ซึ่งขึ้นอยู่กับการเลือกของผู้ใช้งาน

void OnCommit( );

ทำหน้าที่จัดเก็บข้อมูลจากจอภาพลงในแฟ้มข้อมูล

void OnNew( );

ทำหน้าที่เตรียมพื้นที่สำหรับระเบียนใหม่ที่จอภาพและในแฟ้มข้อมูล

void OnSaveFile( );

ทำหน้าที่ปิดแฟ้มข้อมูล

void OnDefInsert( );

ทำหน้าที่เพิ่มคำศัพท์ลงใน ดี-เอส ทรี

void OnNext( );

ทำหน้าที่นำข้อมูลระเบียนถัดไปมาแสดงยังจอภาพ

void OnPrev( );

ทำหน้าที่นำข้อมูลระเบียนก่อนหน้ามาแสดงยังวิว

9. คลาส CFindDlg

บริการของวัตถุ ประกอบด้วย

void OnButSearch( );

มีหน้าที่รับคำศัพท์ที่ผู้ใช้งานป้อนเข้ามา และนำความหมายของคำศัพท์นั้นๆมาแสดง หากค้นหาคำศัพท์แล้วไม่พบ จะแสดงข้อความให้ผู้ใช้งานทราบ

void OnButPaste( );

มีหน้ารับคำศัพท์จากคลิปบอร์ดเพื่อนำมาค้นหาความหมาย

void OnButCopy( );

มีหน้ารับข้อมูลจากจอภาพลงสู่คลิปบอร์ด

void OnButUpdate( );

มีหน้าที่แก้ไขข้อมูลความหมายในแฟ้มข้อมูล

void OnButDelete( );

มีหน้าที่ลบคำศัพท์ที่ไม่ต้องการออกจากพจนานุกรม ซึ่งมีขั้นตอนการทำงาน ดังนี้

1. สืบค้นคำศัพท์โดยใช้ฟังก์ชันสืบค้นคำศัพท์ ถ้าพบคำศัพท์ไปทำงานข้อ 2 ถ้าไม่พบคำศัพท์ลบคำศัพท์ไม่ได้ออกจากฟังก์ชันโดยส่งค่าเท็จ
2. ใส่ค่า ? ในอะเรย์เทล ที่ตำแหน่ง -BASE[r] ซึ่งค่าต่างๆได้มาจากฟังก์ชันการสืบค้นนั่นเอง
3. กำหนดค่า BASE[r] = 0 , CHECK[r] = 0 และ VALUE[r] = 0
4. ออกจากฟังก์ชันโดยส่งค่าจริงกลับ

โดยสรุปแล้วขั้นตอนการลบข้อมูลสามารถกระทำได้โดยตัดความสัมพันธ์ของดับเบิลอะเรย์กับอะเรย์เทล โดยแทนค่าของอะเรย์ที่ตำแหน่งของเซพพาเรทโหนดด้วยศูนย์ และแทนที่ส่วนของซิงเกิลสตริงหรือ STR[r] ในอะเรย์เทลด้วยสัญญลักษณ์การไม่ใช้งาน "?"

10. คลาส CAddDlg

ลักษณะประจำ ประกอบด้วย

int dmaxtype;

int dmaxpron;

int dmaxdef;

int dmaxsyno;

int dmaxanto;

int dmaxabb;

int dmaxsam;

int dmaxdate;

บริการของวัตถุ ประกอบด้วย

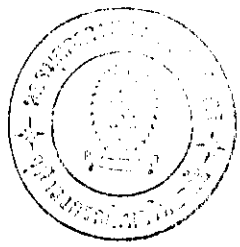
virtual void OnOK();

ทำหน้าที่บันทึกผลการเลือกชนิดข้อมูลที่ต้องการ

11. คลาส CLzw

ลักษณะประจำ ประกอบด้วย

```
struct dictionary {
        int code_value;    ( รหัสของวลีปัจจุบัน )
        int parent_code;   ( รหัสของวลีก่อนหน้า )
        char character;    ( พยัญชนะของโหนดปัจจุบัน )
} *dict[ TABLE_BANKS ];
char    decode_stack[ TABLE_SIZE ];   ( แสตกใช้สำหรับการขยายข้อมูล )
unsigned int next_code;          ( รหัสแรกที่สามารถใช้ได้ )
int     current_code_bits;       ( จำนวนบิต )
unsigned int next_bump_code;     ( ใช้ตรวจสอบเพื่อเพิ่มจำนวนบิต )
```



บริการของวัตถุ ประกอบด้วย

void InitializeDictionary( );

ทำหน้าที่กำหนดค่าเริ่มต้นให้พจนานุกรม เพื่อใช้ทั้งในการบีบอัดและขยายข้อมูล ซึ่งมีขั้นตอนการทำงาน ดังนี้

1. กำหนดค่า code_value ทั้งหมดในพจนานุกรมให้มีค่าเท่ากับ -1
2. กำหนดค่า next_code เท่ากับ 259
3. กำหนดค่า current_code_bits เท่ากับ 9 (เนื่องจากในตอนเริ่มต้นจะใช้บิตขนาด 9 บิต)
4. กำหนดค่า next_bump_code เท่ากับ 511

void InitializeStorage( );

ทำหน้าที่จองเนื้อที่สำหรับกลุ่มของตัวชี้ของ dict

void CompressFile(FILE *input, BIT_FILE *output);

ทำหน้าที่อ่านข้อมูลจากแฟ้มข้อมูล เพื่อนำมาทำการลดขนาดข้อมูลและบันทึกลงแฟ้มข้อมูลที่ผ่านการบีบอัดข้อมูลแล้ว ซึ่งมีขั้นตอนการทำงาน ดังนี้

1. กำหนดค่า code_value ให้ทุกโหนดในอะเรย์มีค่าเป็น -1
2. อ่านตัวอักษรจากแฟ้มข้อมูลเข้ามา และกำหนดค่าให้ string_code ในขณะเดียวกันก็ตรวจสอบด้วยว่าอักษรที่อ่านเข้ามานั้นเป็นตัวชี้สิ้นสุดแฟ้มข้อมูลหรือไม่ ถ้าใช่กำหนดค่าให้ string_code เท่ากับ END_OF_STREAM ในที่นี้กำหนดให้เป็นรหัส256 ถ้าไม่ใช่ทำข้อ 3
3. อ่านตัวอักษรจากแฟ้มข้อมูลเข้ามา และกำหนดค่าให้ character จนกว่าจะพบตัวชี้สิ้นสุดแฟ้มข้อมูล
4. ตรวจสอบว่า string_code มีโหนดลูกหรือไม่ โดยใช้ฟังก์ชัน find_child_node

   ถ้าพบ กำหนดค่า string_code ให้เท่ากับ code_value ของโหนดลูก และกลับไปทำ ข้อ 3

   ถ้าไม่พบคืนค่า string_code และเพิ่มรหัสใหม่ในอะเรย์ โดยกำหนดให้

   code_value เท่ากับ next_code

   parent_code เท่ากับ string_code

   character เท่ากับ character

จากนั้นกำหนดค่า string_code ใหม่ โดยให้เท่ากับ character และกลับไปทำข้อ 3

void ExpandFile(BIT_FILE *input, FILE *output);

ทำหน้าที่ในการขยายข้อมูลจากแฟ้มข้อมูลที่ผ่านการบีบอัดข้อมูล เปลี่ยนรหัสเป็นวลีและเขียนลงแฟ้มข้อมูลส่งออก ซึ่งมีขั้นตอนการทำงาน ดังนี้

1. อ่านรหัสจากแฟ้มข้อมูล และกำหนดค่าให้ old_code
2. ตรวจสอบถ้า old_code เท่ากับ ตัวชี้สิ้นสุดแฟ้มข้อมูล โปรแกรมสิ้นสุดการทำงาน
3. กำหนดค่า character เท่ากับ old_code และส่งค่า old_code เป็นเอาต์พุต
4. อ่านรหัสจากแฟ้มข้อมูล และกำหนดค่าให้ new_code จนกระทั่งพบตัวชี้สิ้นสุดแฟ้มข้อมูล
5. ตรวจสอบว่า new_code มากกว่าหรือเท่ากับ next_code

   ถ้าใช่ กำหนดค่า decode_stack[0] เท่ากับ character

   หาค่า count (จำนวนตัวอักษรที่อยู่ในสแตก) โดยใช้ฟังก์ชัน decode_string

   ถ้าไม่ใช่ หาค่า count โดยใช้ฟังก์ชัน decode_string
6. กำหนดค่า character ให้เท่ากับ decode_stack [count -1]
7. ส่งค่าในสแตกออกเป็นเอาต์พุต ตั้งแต่ค่า decode_stack[–count] จนกระทั่ง count น้อยกว่าหรือเท่ากับ 0
8. ตรวจสอบว่า next_code น้อยกว่า หรือเท่ากับ MAX_CODE

   ถ้าใช่ เพิ่มรหัสใหม่ในพจนานุกรม โดยกำหนดค่า parent_code เท่ากับ old_code และ character

   เท่ากับ character ต่อไปเพิ่มค่าให้ next_code อีก 1
9. กำหนดค่า old_code เท่ากับ new_code และกลับไปทำข้อ 4

unsigned int find_child_node(int parent_code, int child_character);

ฟังก์ชันนี้ เป็นแฮชชิงฟังก์ชัน เพื่อหาตำแหน่งที่สอดคล้องกับค่าของ parent_code รวมกับค่าของ child_character ซึ่งมีขั้นตอนการทำงาน ดังนี้

1. หาค่า index ซึ่งสอดคล้องกับค่า parent_code และ child_character .
2. ตรวจสอบค่า index ว่าเท่ากับ 0 หรือไม่

   ถ้าใช่ กำหนดค่าออฟเซ็ท เท่ากับ 1

   ถ้าไม่ใช่ กำหนดค่าออฟเซ็ท เท่ากับ TABLE_SIZE - index
3. ตรวจสอบค่า code_value ที่ตำแหน่งเท่ากับ index ว่าเท่ากับ -1 หรือไม่

ถ้าใช่ return ค่า index

4. ตรวจสอบค่า parent_code ที่ตำแหน่งเท่ากับ index ว่าเท่ากับ ค่า parent_code และค่า character ที่ตำแหน่งเท่ากับ index เท่ากับ child_character หรือไม่

ถ้าใช่ return ค่า index

5. ตรวจสอบว่า index >= offset หรือไม่

ถ้าใช่ กำหนดค่า index เท่ากับ index - offset

ถ้าไม่ใช่ กำหนดค่า index เท่ากับ index + (TABLE_SIZE - offset)

6. กลับไปทำข้อ 3

unsigned int decode_string(unsigned int count, unsigned int code);

ใช้สำหรับการขยายข้อมูล เนื่องจากการขยายข้อมูลจะทำกลับกับการบีบอัดข้อมูล จึงต้องนำข้อมูลที่ผ่านการถอดรหัสแล้วบันทึกลงแสตก และจะคืนค่าจำนวนอักขระที่อยู่ในสแตก

มีขั้นตอนการทำงาน ดังนี้

1. ตรวจสอบว่า code มากกว่า 255 หรือไม่

ถ้าใช่ ทำข้อ 2

2. นำค่า character ใส่ลงในสแตก
3. กำหนดค่า code เท่ากับ parent_code
4. กลับไปทำข้อ 1
5. กำหนดค่า decode_stack เท่ากับ code
6. คืนค่าจำนวนตัวอักษรที่อยู่ในสแตก

เนื่องจากในการบีบอัดเพื่อลดขนาดข้อมูล จะต้องอ่านและเขียนแฟ้มข้อมูลในลักษณะบิต การอ่านและเขียนแฟ้มข้อมูลในลักษณะบิต ประกอบด้วยโครงสร้าง ดังนี้

```
typedef struct bit_file{
        FILE *file;
        unsigned char mask;
        int rack;
        int pacifier_counter;
}BIT_FILE;
```

rack	บรรจุข้อมูลที่อ่านเข้ามาหรือที่จะบันทึกลงแฟ้มข้อมูลทีละไบต์
mask	กำหนดค่าเริ่มต้นเป็น 0x80 ใช้เป็นตัวเลื่อนลำดับการอ่านค่าจาก rack มาทีละบิต จนกว่าจะครบทั้งไบต์ จากนั้นจะอ่านค่าไบต์ต่อไปเข้าไปเก็บใน rack
pacifier_counter	ใช้ตรวจสอบความถูกต้องในการอ่านและเขียนแฟ้มข้อมูล โดยจะเพิ่มค่าทีละหนึ่งเมื่อมีการ อ่านไบต์ใหม่เข้ามา หรือบันทึกลงแฟ้มข้อมูล

ประกอบด้วยฟังก์ชัน ดังนี้

```
BIT_FILE        *OpenInputBitFile( char *name );
BIT_FILE        *OpenOutputBitFile( char *name );
void            OutputBits( BIT_FILE *bit_fiel, int bit );
unsigned long   InputBits( BIT_FILE *bit_file, int bit_count );
void            CloseInputBitFile( BIT_FILE *bit_file );
void            CloseOutputBitFile( BIT_FILE *bit_file );
```

การทำงานของแต่ละฟังก์ชัน มีดังนี้

ฟังก์ชัน OpenInputBitFile และ OpenOutputBitFile

ทำหน้าที่เปิดแฟ้มข้อมูล จองเนื้อที่หน่วยความจำให้ BITFILE structure กำหนดค่า rack = 0 , mask = 0x80, pacifier_counter = 0 และ return file structure pointer ซึ่งชี้ไปยัง structure ใหม่

ฟังก์ชัน InputBits

ฟังก์ชันนี้ทำหน้าที่อ่านข้อมูลจากแฟ้มข้อมูลเข้ามา และส่งค่าออกมาทีละบิต

1. ตรวจสอบว่า mask เท่ากับ 0x80 หรือไม่

   ถ้าใช่ ทำข้อ 2

   ถ้าไม่ใช่ ทำข้อ 3

2. อ่านข้อมูลจากแฟ้มข้อมูลเข้ามาเก็บใน rack ทีละไบต์
3. อ่านข้อมูลจาก rack เข้ามาเก็บใน mask ทีละบิต
4. ตรวจสอบว่า rack บิท-แอนด์ กับ mask เท่ากับ 0 ทั้งหมดหรือไม่

   ถ้าใช่ กำหนดค่า return_value เท่ากับ บิท-ออร์ ของ mask

5. ตรวจสอบว่า mask เท่ากับ 0 หรือไม่

   ถ้าใช่ กำหนดค่าให้ mask เท่ากับ 0x80

6. กลับไปทำข้อ 2 จนกว่า จะพบว่า mask เท่ากับ 0

ฟังก์ชัน OutputBits

ฟังก์ชันนี้ ทำหน้าที่ในการเขียนข้อมูลเอาต์พุท

1. กำหนดค่าให้กับ mask เท่ากับ การเลื่อนบิตของ bit_count -1 ไปทางซ้าย 1L
2. ตรวจสอบว่า mask ไม่เท่ากับ 0

   ถ้าใช่ ทำข้อ 3
3. ตรวจสอบว่า ค่าของ mask บิท-แอนด์ กับค่า code เท่ากับ 0 ทั้งหมดหรือไม่

   ถ้าใช่ กำหนดค่าของ rack เท่ากับ บิต-ออร์ ของ mask
4. เลื่อนบิตของ mask ไปทางขวา 1 บิต
5. ตรวจสอบว่า mask เท่ากับ 0 หรือไม่

   ถ้าใช่ ทำข้อ 6
6. ส่งค่าใน rack สู่เอาต์พุต
7. กำหนดค่า rack เท่ากับ 0 และ mask เท่ากับ 0x80
8. เลื่อนบิตของ mask ไปทางขวา 1 บิต
9. กลับไปทำข้อ 2