



โปรแกรมการจัดการและการออกแบบแบบฟอร์ม

Form Generate And Administer

โดย

นายชาญฤทธิ์ ดวงสุวรรณ

นายอนุวัฒน์ มนุรัตน์ศิลป์

อาจารย์ที่ปรึกษา

ดร. ชม กิมปาน

วัน เดือน ปี... ๑๑ ก.ค. ๒๕๕๐

เลขทะเบียน... ๐๖๓๐๖๕

เลขเรียกหนังสือ... T 38458- ๔ 481 ป.

ปริญญานิพนธ์สำหรับปริญญาวិชากรรมศาสตรบัณฑิต

สาขาวิชาวิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์

สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

ปีการศึกษา 2538

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ปริญญานิพนธ์ปีการศึกษา 2538

ภาควิชา วิศวกรรมคอมพิวเตอร์

คณะวิศวกรรมศาสตร์ สถาบันเทคโนโลยีพระจอมเกล้าเจ้าคุณทหารลาดกระบัง

เรื่อง โปรแกรมการจัดการและการออกแบบแบบฟอร์ม

ผู้จัดทำ

1. นายชาญฤทธิ์ ควขสุวรรณ 36.013148

2. นายอนุวัฒน์ มนุรัตน์ศิลป์ 36.013185



โปรแกรมการจัดการและออกแบบแบบฟอร์ม

นายชาญฤทธิ์ ดวงสุวรรณ

นายอนุวัฒน์ มนุรัตน์ศิลป์

ค.ร. ชม กิมปาน อาจารย์ที่ปรึกษา

ปีการศึกษา 2538

บทคัดย่อ

โดยทั่วไปแบบฟอร์มที่เราใช้กัน จะมีอยู่หลายแบบ และเรายังต้องเขียน(กรอก)ข้อความต่าง ๆ ด้วยลายมือเราเอง กล่าวถึงความรวดเร็ว ความสวยงาม อีกทั้งความถูกต้องอาจยังไม่เพียงพอ โปรแกรมสร้างและกรอกแบบฟอร์มสำหรับงานธุรการนี้ ได้สร้างขึ้นเพื่อเป็นเครื่องมือช่วยในการกรอกแบบฟอร์ม โดยคำนึงถึงความสะดวกในการใช้งาน ความรวดเร็ว ความสวยงาม และความถูกต้องเป็นหลัก โดยการทำงานในครั้งแรก จะต้องสร้างแบบฟอร์มที่มีลักษณะและตำแหน่งเหมือนกับแบบฟอร์มที่จะใช้กรอก โดยจะใช้วิธีลากแล้ววาง ซึ่งเราจะมีอุปกรณ์ต่าง ๆ ให้ใช้เพียงพอที่จะสร้างแบบฟอร์มให้ได้เหมือนจริงมากที่สุด ซึ่งแบบฟอร์มนี้เราสามารถเก็บบันทึกไว้ใช้งานได้ในครั้งต่อไปด้วย โดยใช้โปรแกรมสร้างแบบฟอร์ม จากนั้นจึงทำการกรอกข้อความลงไปบนแบบฟอร์ม แล้วพิมพ์ออกทางเครื่องพิมพ์ ซึ่งจะพิมพ์ลงบนแบบฟอร์มตัวจริงที่ป้อนเข้าไป โดยสามารถพิมพ์ได้ตรงตำแหน่ง ในความละเอียดถึง 1 มิลลิเมตร

Form Generate And Administrator

Chanrit Doungsuwan

Anuwat Manurattanasilp

Chom Kimpan Advisor

1995

Abstract

General Forms which we use have many type and we still use handwritten. So about speed, tidy and correctness are not adequate.

This program form generate and administration had built for help our fill form. Depend on user-friendly, speed, tidy and correctness mainly. In process, first we will created form that has characteristic and position in the same as original form. By method drag and drop object which has many to use. Form which we created can save for use in the next time. by keyboard are can change font and size of text then print it to printer on the original form and can print accurately precise in 1mm.

๕

บทคัดย่อ

ABSTRACT

สารบัญ

สารบัญรูปภาพ

สารบัญตาราง

บทที่ 1 บทนำ

1.1 ที่มาของปัญหา 1-1

1.2 วัตถุประสงค์ของปริญญานิพนธ์ 1-2

บทที่ 2 ทฤษฎีและหลักการ

2.1 คอมไพเลอร์และอินเตอร์พรีเตอร์ 2-1

2.2 ตัวแปรและค่าคงตัว 2-3

2.3 ฟังก์ชัน 2-5

2.5 การสร้างตัวแปรและค่าคงตัว 2-6

2.6 ข้อความสั่งนิพจน์และกระบวนการ 2-8

2.7 นิพจน์และโอเปอเรเตอร์ 2-9

2.8 ชนิดของนิพจน์ 2-10

2.9 การทำก่อนของโอเปอเรเตอร์คำนวณ 2-14

2.10 โอเปอเรเตอร์เชื่อมต่อสตริง 2-16

2.11 โอเปอเรเตอร์เปรียบเทียบ 2-16

2.12 โอเปอเรเตอร์ตรรกะ 2-18

2.13 แบบฟอร์ม 2-23

2.14 เทเบิล 2-30

2.15 ปุ่มคำสั่ง เช็kb็อกซ์ และออปชัน 2-34

2.16 กรอบ 2-36

2.17 ลิสต์บ็อกซ์ 2-37

2.18 คอมโบบ็อกซ์ 2-40

2.19 ตัวจับเวลา 2-44

2.20	ใคร่พิลิตส์บ็อกซ์ ใคร่กทอริลิตส์บ็อกซ์ และฟลลิตส์บ็อกซ์	2-48
2.21	กริด	2-50
2.22	คอมมอนไดอะล็อกซ์บ็อกซ์	2-51
2.22.1	Open	2-52
2.22.2	Save As	2-54
2.22.3	Color	2-55
2.22.4	Font	2-56
2.22.5	Print	2-57
2.22.6	Help	2-58
2.23	เมนู	2-58
2.24	โครงสร้างฐานข้อมูล	2-70
2.25	โครงสร้างฐานข้อมูลใน JET 1.1	2-72
2.25.1	ฐานข้อมูล	2-72
2.25.2	ครรรชนี	2-73
2.26	การจัดการโครงสร้างฐานข้อมูล	2-75
2.27	ชุดคำสั่งสำหรับการจัดการโครงสร้างฐานข้อมูล	2-77
2.28	ตัวเลือกในการกำหนดลักษณะของฐานข้อมูล	2-82
2.29	การใช้วิซวลเบสิคกับฐานข้อมูลอื่น ๆ	2-86
2.30	การใช้วิซวลเบสิคกับฐานข้อมูล ODBC	2-89
2.31	การรับและการส่งออกผ่านไฟล์	2-92
2.32	ลักษณะทั่วไปของระบบไฟล์	2-92
2.33	การดำเนินการกับไฟล์	2-93
2.34	ไฟล์ลำดับ	2-94
2.34.1	การเปิดและปิดไฟล์ลำดับ	2-94
2.34.2	การอ่านจากไฟล์ลำดับ	2-95
2.34.3	การเขียนไปยังไฟล์ลำดับ	2-96
2.35	ไฟล์เข้าถึงเชิงสุ่ม	2-97
2.35.1	การเปิดและการปิดไฟล์เข้าถึงเชิงสุ่ม	2-98
2.35.2	การอ่านและการเขียนไปยังไฟล์เข้าถึงเชิงสุ่ม	2-100
2.36	ไดนามิกลิงค์ไลบรารี	2-104

บทที่ 3 การสร้าง

3.1 การออกแบบให้โปรแกรมทำงานภายใต้สิ่งแวดล้อม Windows	3-1
3.2 เลือกเครื่องมือในการพัฒนา	3-1
3.3 การออกแบบ การทำงานและการติดต่อกับผู้ใช้	3-1
3.4 รายละเอียดการทำงานของโปรแกรม	3-5
3.4.1 Genform	3-5
3.4.2 Color Form	3-7
3.4.3 Form	3-8
3.4.4 Helptip	3-13
3.4.5 Linkdb	3-13
3.4.6 Main	3-14
3.4.7 Newform	3-15
3.4.8 Property	3-15
3.4.9 Tool	3-18
3.4.10 Formsize	3-18
3.4.11 Global.bas	3-19
3.4.12 Helptip.bas	3-20

บทที่ 4 การทดลองและผลการทดลอง

4.1การทดลองเกี่ยวกับภาพลักษณ์	4-1
4.2 การทดลองเกี่ยวกับ Text Box	4-4
4.3 การทดลองเกี่ยวกับ Option, Check Box and Command Button	4-6
4.4 การทดลองเกี่ยวกับเฟรม	4-8
4.5 การทดลองเกี่ยวกับ List Box	4-11
4.6 การทดลองเกี่ยวกับ Combo Box	4-12
4.7 การทดลองเกี่ยวกับ Scroll Bar	4-15
4.18 การทดลองเกี่ยวกับ Timer	4-17
4.19 การทดลองเกี่ยวกับ Shape, Image and Line	4-18
4.20 การทดลองเกี่ยวกับ File	4-20
4.21 การทดลองเกี่ยวกับ Sequential File	4-27

4.22 การทดลองเกี่ยวกับ Random File

4-31

บทที่ 5 วิจัยและสรุป

5.1 สรุปย่อ

5-1

หนังสืออ้างอิง



2.1 แปลภาษาต้นฉบับไปเป็นภาษาเครื่อง	2-1
2.2 การคอมไพล์และการอินเตอร์พรีต	2-2
2.3 การสูญเสียสารสนเทศเป็นผลของโอเวอร์โฟลว์	2-5
2.4 การสวนวนหน่วยเก็บโดยอาศัยการประกาศตัวแปร	2-6
2.5 การใช้โอเปอเรเตอร์หารจำนวนเต็ม	2-13
2.6 การใช้โอเปอเรเตอร์หารจำนวนเต็มและโอเปอเรเตอร์ Mod	2-14
2.7 เครื่องมือสร้างบ็อกซ์ภาพ	2-29
2.8 เครื่องมือสร้างภาพลักษณะ	2-29
2.9 ชั้นของจอแสดงผลแบบฟอร์ม	2-30
2.10 เครื่องมือสร้างเลเบล	2-31
2.11 เครื่องมือสร้างเท็กซ์บ็อกซ์	2-32
2.12 เครื่องมือสร้างปุ่มคำสั่ง	2-35
2.13 เครื่องมือสร้างเช็ทซ์บ็อกซ์	2-35
2.14 เครื่องมือสร้างปุ่มออพชัน	2-35
2.15 เครื่องมือสร้างกรอบ	2-36
2.16 เครื่องมือสร้างลิสต์บ็อกซ์	2-38
2.17 เครื่องมือสร้างคอม โบบ็อกซ์	2-40
2.18 สกรอลล์บาร์ตามแนวนอนและแนวตั้ง	2-42
2.19 เครื่องมือสกรอลล์บาร์ตามแนวนอน	2-43
2.20 เครื่องมือสกรอลล์บาร์ตามแนวตั้ง	2-43
2.21 ตัวจับเวลา	2-44
2.22 เครื่องมือสร้างเส้น	2-46
2.23 เครื่องมือสร้างรูปทรง	2-46
2.24 เครื่องมือสร้างโครฟลิสต์บ็อกซ์	2-48
2.25 เครื่องมือสร้างโครเรททอรีลิสต์บ็อกซ์	2-48
2.26 เครื่องมือสร้างไฟล์ลิสต์บ็อกซ์	2-48
2.27 เครื่องมือสร้างกริด	2-50
2.28 เครื่องมือสร้างข้อมูล	2-51
2.29 เครื่องมือสร้างคอมมอน ไดอะล๊อกซ์บ็อกซ์	2-52
2.30 ไดอะล๊อกซ์ Open	2-53

2.31	ไดอะล็อกซ์ Save As	2-54
2.32	ไดอะล็อกซ์บ็อกซ์ Color	2-55
2.33	ไดอะล็อกซ์ Font	2-56
2.34	ไดอะล็อกซ์บ็อกซ์ Print	2-57
2.35	แอปพลิเคชัน Help	2-58
2.36	วินโดว์ออกแบบเมนูของ Visual Basic	2-59
2.37	แสดงหน้าต่างสร้างเมนู	2-63
2.38	แบบฟอร์มกับเมนูบาร์และเมนูแบบดิ่งลง	2-64
2.39	เมนูพร้อมกับมีแท่งแยก เครื่องหมาย และอุปชันสภาพใช้การไม่ได้	2-67
2-40	ป๊อปอัพเมนู	2-68
2.41	ตัวอย่างตารางข้อมูลพนักงาน	2-69
2.42	ตัวอย่างตารางข้อมูลหน่วยงาน	2-70
2.43	โครงสร้างและส่วนประกอบของฐานข้อมูลใน JET 1.1	2-73
2.44	แสดงวินดอร์ของโปรแกรม Data Manager	2-75
2.45	แสดงส่วนประกอบต่าง ๆ ของฐานข้อมูล	2-78
2.46	ไฟล์ข้อมูลที่มีสี่เรคอร์ดสำหรับฐานข้อมูล Weasel	2-103
2.49	เครื่องมือสร้างไฟล์, ไครเททอรี, ไครฟลิสค์บ็อกซ์	2-93
2.50	รูปแสดงฐานข้อมูลของ Weasel	2-103
3.1	แสดงโปรแกรมขณะทำงาน	3-2
3.2	แสดงหน้าต่างหลักขณะออกแบบสร้าง	3-2
3.3	แสดงหน้าต่างออกแบบขณะออกแบบ	3-3
3.4	แสดงหน้าต่างเครื่องมือขณะออกแบบ	3-4
3.5	แสดงหน้าต่างคุณสมบัติขณะออกแบบ	3-4
3-6	แสดงหน้าต่างเลือกสีขณะออกแบบ	3-5
3-7	แสดงหน้าต่างแสดง Tips ขณะออกแบบ	3-5
4.1	ภาพประกอบการทดลองภาพลักษณ์	4-1
4.2	ภาพประกอบการทดลองภาพลักษณ์	4-2
4.3	ภาพประกอบการทดลองภาพลักษณ์	4-2
4.4	ภาพประกอบการทดลองภาพลักษณ์	4-4
4.5	ภาพประกอบการทดลอง Text Box	4-5

4.6 ภาพประกอบการทดลอง Text Box	4-6
4.7 ภาพประกอบการทดลอง Option, Check Box and Command Button	4-6
4.8 การอินเตอร์เฟซของ Game	4-7
4.9 ภาพประกอบการทดลองเฟรม	4-9
4.10 อินเตอร์เฟซของ Game ที่ได้รับการออกแบบใหม่	4-9
4.11 แบบฟอร์มที่ออกแบบไว้สำหรับแอปพลิเคชัน Cheese	4-10
4.12 รหัสโปรแกรมสำหรับแอปพลิเคชัน Cheese	4-11
4.13 ภาพประกอบการทดลอง Combo Box	4-12
4.14 ภาพประกอบการทดลอง Combo Box	4-13
4.15 รหัสโปรแกรมสำหรับแอปพลิเคชัน Flight	4-14
4.16 การดำเนินงานแอปพลิเคชัน Flight	4-14
4.17 ภาพประกอบการทดลอง Scroll Bar	4-15
4.18 การดำเนินงานแอปพลิเคชัน Scroll Bar	4-16
4.19 ภาพประกอบการทดลอง Timer	4-17
4.20 ภาพประกอบการทดลอง Shape, Image and Line	4-19
4.21 ภาพประกอบการทดลอง Shape, Image and Line	4-19
4.22 ภาพประกอบการทดลอง Shape, Image and Line	4-20
4.23 การออกแบบฟอร์มสำหรับไดอะล็อกบ็อกซ์ Open File	4-22
4.24 ไดอะล็อกบ็อกซ์ Project Options	4-23
4.25 รหัสโปรแกรมสำหรับกระบวนการงานอีเวนต์ใน Application TestFile	4-26
4.26 การออกแบบแบบฟอร์มตอนแรกเริ่มสำหรับแอปพลิเคชัน Configurator	4-27
4.27 รหัสโปรแกรมสำหรับแอปพลิเคชัน Configurator	4-29
4.28 การแก้ไขไฟล์ Config.sys ด้วยแอปพลิเคชัน Configurator	4-30
4.29 การออกแบบฟอร์มแรกเริ่มสำหรับแอปพลิเคชันฐานข้อมูล Weasel	4-32
4.30 การออกแบบฟอร์มที่สมบูรณ์สำหรับแอปพลิเคชันฐานข้อมูล Weasel	4-36
4.31 กระบวนการ ShowWeasel และ Update CurrentWeasel	4-38
4.32 กระบวนการฟังก์ชัน NewWeasel	4-39
4.33 กระบวนการอีเวนต์สำหรับเท็กบ็อกซ์ในแอปพลิเคชันฐานข้อมูล Weasel	4-42
4.34 การดำเนินงานแอปพลิเคชันฐานข้อมูล Weasel	4-43

2.1 ชนิดข้อมูลของ Visual Basic	2-4
2.2 อักขระประกาศของ Visual Basic	2-8
2.3 โอเปอเรเตอร์คำนวณ	2-11
2.4 ตัวอย่างของการยกกำลัง	2-12
2.5 การตัดและการปิดเศษ	2-13
2.6 โอเปอเรเตอร์คำนวณในลำดับของการทำก่อน	2-15
2.7 การใช้กฎการคำนวณก่อนเพื่อคำนวณค่า	2-15
2.8 โอเปอเรเตอร์เปรียบเทียบ	2-17
2.9 การใช้โอเปอเรเตอร์เปรียบเทียบ	2-17
2.10 โอเปอเรเตอร์ตรรกะหรือ Boolean เรียงลำดับตามก่อนหลัง	2-21
2.11 การใช้โอเปอเรเตอร์ตรรกะหลาย ๆ ตัวในนิพจน์หนึ่ง ๆ	2-21
2.12 โอเปอเรเตอร์ของ Visual Basic เรียงลำดับตามการทำก่อน	2-22
2.13 เรคอร์ดในไฟล์เข้าถึงเชิงสุ่ม	2-100
4.1 การตั้งคุณสมบัติสำหรับแอปพลิเคชัน Game	4-7
4.2 การตั้งคุณสมบัติสำหรับแบบฟอร์มตัวที่สองในแอปพลิเคชัน TestFile	4-22
4.3 การตั้งคุณสมบัติสำหรับแอปพลิเคชัน Configurator	4-29
4.4 การตั้งคุณสมบัติแอปพลิเคชันฐานข้อมูล Weasel	4-34

บทที่ 1 บทนำ

ความก้าวหน้าทางเทคโนโลยี ด้านคอมพิวเตอร์รุดหน้าไปอย่างรวดเร็ว ประกอบกับราคาคอมพิวเตอร์ถูกลงและโปรแกรม Microsoft Windows ได้มีการใช้กันอย่างแพร่หลาย โดยมีโปรแกรมรองรับมากมายทั้งในด้านวิศวกรรม ด้านสถาปัตยกรรม ด้านธุรกิจ ฯลฯ เป็นผลให้มีการนำคอมพิวเตอร์มาใช้งานมากขึ้น แต่ไม่ว่าจะเป็นงานด้านใด คงปฏิเสธไม่ได้ว่าจะต้องมีการใช้ฟอร์มเอกสาร ช่วยในการนำเสนอข้อมูล ซึ่งมีโปรแกรมสำเร็จรูปทั่ว ๆ ไป มาใช้งานพิมพ์เอกสารที่เป็นแบบฟอร์ม เช่น LOTUS, dBASE, WORD PROCESSOR อื่น ๆ แต่ก็ยังมีปัญหามากมาย เช่น ไม่สามารถพิมพ์ข้อความให้ตรงกับช่องได้ หรือให้ตรงบรรทัดของแบบฟอร์มได้, ไม่สามารถพิมพ์ตัวอักษรในขนาดที่พอเหมาะกับขนาดของช่องว่างได้, ภายใน 1 ฟอร์มไม่สามารถเปลี่ยนขนาดหรือรูปแบบตัวอักษรได้, การกรอกฟอร์มซึ่งต้องเลื่อนและกระชากของ CURSOR ในการเปลี่ยนตำแหน่ง ไม่ตรงหรือต้องปรับทุกครั้งที่เปลี่ยน FIELD, การพิมพ์ออก PRINTER ซึ่ง APPLICATION SOFTWARE บางตัวไม่ SUPPORT กับ PRINTER หรือเมื่อมีการผลิต PRINTER ออกมาจำหน่ายก็จะมีเพียง DRIVER ของโปรแกรม WINDOWS ให้มาเท่านั้น หรือต้องมีการ SET FORM ใหม่เมื่อมีการนำ PRINTER ตัวอื่นมาใช้งาน ซึ่ง PROJECT นี้จะแก้ปัญหาของการพิมพ์ฟอร์มได้เป็นอย่างดี เช่น ช่วยในการเลื่อน CURSOR การจัดรูปแบบหรือขนาดของตัวอักษรให้มีความเหมาะสมกับแบบฟอร์ม หรือนำข้อมูลจาก ฐานข้อมูลมากรอกลงฟอร์มโดยอัตโนมัติ และโปรแกรมนี้ทำงานภายใต้ WINDOWS ซึ่งจะช่วยแก้ปัญหาของการพิมพ์ฟอร์มกับเครื่องพิมพ์ได้เป็นอย่างดี

ที่มาของปัญหา

1. ในปัจจุบันแบบฟอร์มที่ใช้กันอยู่นั้น ยังต้องใช้คนกรอก (เขียน) ลงไปอยู่ อีกทั้งมีรูปแบบมากมาย ถ้าต้องการกรอกเป็นจำนวนน้อยก็คงไม่เป็นปัญหานัก แต่ถ้าต้องการกรอกเป็นจำนวนมาก ก็จะเป็นปัญหาและเกิดการล่าช้า และไม่สะดวกนัก เช่น อาจต้องการกรอกแบบฟอร์มแบบหนึ่งเป็นจำนวน 20 แผ่น และต้องเปลี่ยนแคปซูล ชื่อ-นามสกุล ส่วนฟิลด์อื่น ๆ ยังคงเดิม แต่ถ้ามีซอฟต์แวร์ที่ช่วยในการกรอก เราก็จะทำได้รวดเร็วขึ้นและสะดวกกว่า เพราะเราสามารถบันทึก และสามารถเปิดขึ้นมาแก้ไขได้ ดังนั้น เราก็จะแก้ไขเฉพาะส่วนที่ต้องการ มีการคัดลอกและตัดปะ เหมือนที่มีใน โปรแกรมประมวลผลคำ (Word Processor) ทั่วไป หรือ ต้องการกรอกแบบฟอร์มเหมือนกันทุกประการหลาย ๆ ชุด ถ้าเป็นการกรอกแบบธรรมดาก็ต้องกรอกเป็นจำนวนเท่ากับชุดที่ต้องการ บางคนอาจจะแย้งว่า ก็กรอกเพียงแค่ว่าชุดเดียว แล้วก็ถ่ายเอกสารเอาก็ได้ ก็อาจจะเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่นิยมนำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งยังมีให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ทำได้ถ้าไม่ได้เป็นแบบฟอร์มเฉพาะที่ต้องกรอกบนแบบฟอร์มที่ให้มาเท่านั้น แต่ซอฟต์แวร์ตัวนี้สามารถทำได้โดยสะดวกด้วยการสั่งพิมพ์ครั้งเดียว เพราะเราจะพิมพ์ลงบน Preprint Form ที่ให้มา

2. ในการกรอกแบบธรรมดาถ้าเรากรอกผิด ก็จะต้องลบหรือขีดฆ่าทิ้ง ถ้าแบบฟอร์มนั้นห้ามให้ทำการแก้ไขหรือรอยขีดขีด เราก็ต้องเอาแผ่นใหม่มากรอก แต่ถ้าเรามีซอฟต์แวร์ที่มาช่วยในการกรอกทำให้เราสามารถ ตรวจสอบแก้ไข ก่อนที่จะพิมพ์ลงบนฟอร์มจริง ๆ ได้ประหยัดทั้งเวลาที่ต้องกรอกแผ่นใหม่และจำนวนแบบฟอร์มที่ต้องใช้

3. ในการกรอกแบบธรรมดา(เขียนลงไปเอง) ลายมือของเราอาจจะดูไม่สวยงาม ไม่เป็นระเบียบเรียบร้อย แต่ถ้าใช้ คอมพิวเตอร์กรอกเราสามารถที่จะเลือกรูปแบบตัวอักษรได้ มีความสวยงาม เป็นระเบียบเรียบร้อย อีกทั้งยังเลือกขนาดได้เหมาะสม เลือกเป็นตัวหนา ตัวเอียงได้

4. ถ้าเรามีระบบคอมพิวเตอร์ใช้งานอยู่ แล้วมีระบบฐานข้อมูลใช้อยู่ด้วย แล้วต้องการนำข้อมูลที่มีอยู่ในฐานข้อมูลมากรอกใส่ในแบบฟอร์ม ถ้าใช้การกรอกด้วยคน ก็อาจใช้วิธี พิมพ์ข้อมูลที่ต้องการออกมานกกระดาษก่อน แล้วจึงมาอ่าน หรืออ่านบนหน้าจอ ที่ละรายการ แล้วจึงทำการกรอก(เขียน)ลงบนแบบฟอร์มไปที่ละรายการ แต่ถ้าเรามีซอฟต์แวร์มาช่วยในการ กรอก เราสามารถนำข้อมูลที่มีอยู่ในฐานข้อมูลมากรอกแบบฟอร์มได้

5. ในการพิมพ์แบบฟอร์มให้ตรงฟอร์มนั้นเป็นปัญหากับทุก ๆ องค์การ เพราะ APPLICATION SOFTWARE ไม่สามารถตอบสนองความต้องการได้ทั้งหมด สำหรับโปรแกรมนี้มีความละเอียดในการปรับระยะห่างระหว่างบรรทัดได้ละเอียดมาก (ไม่น้อยกว่า 0.5 มิลลิเมตร)

6. การกรอกแบบฟอร์มนั้นมีความล่าช้าเมื่อใช้ APPLICATION SOFTWARE เช่น ต้องมีการเลื่อน CURSOR ให้ตรงตำแหน่งก่อนพิมพ์ แต่โปรแกรมนี้ได้ทำให้การกรอกข้อความมีความสะดวกและรวดเร็วขึ้น เพียงกด Tab ก็จะไป CURSOR ไปที่ตำแหน่งที่จะกรอกข้อความถัดไป

7.การกรอกข้อมูลมีความผิดพลาดได้ง่ายสำหรับข้อมูลที่มีการคำนวณ เช่นการหาค่าเฉลี่ยของตัวเลข หรือการหาผลรวมของยอดเงินหรืออาจจะกลายเป็นความล่าช้าในการกรอกข้อมูล เพราะต้องนำเอาเครื่องคำนวณมาคิดหาค่าก่อนกรอกข้อมูลลงไป

8. APPLICATION SOFTWARE บางตัวที่ใช้อยู่ไม่สามารถรองรับเครื่องพิมพ์ที่ออกจำหน่ายใหม่ได้ เช่น CU WRITER ใช้พิมพ์กับ HP 560C ไม่ได้ เพราะไม่มี DRIVER ให้เลือก และที่ตัวเครื่องพิมพ์ก็ไม่มีให้มา

วัตถุประสงค์ของโครงการ

โครงการนี้ได้จัดทำขึ้นเพื่อแก้ปัญหาในการกรอกแบบฟอร์มต่าง ๆ ให้มีความสะดวก รวดเร็วและมีความถูกต้องมากขึ้น โดยแบ่งเป็นรายละเอียดย่อย ๆ คือ เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

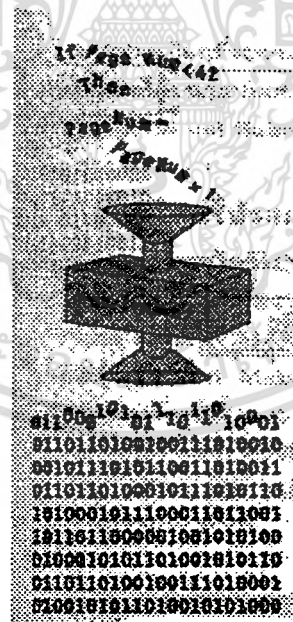
1. เพื่อช่วยให้งานออกเอกสารที่เกี่ยวกับแบบฟอร์มมีความถูกต้องเพราะจะมีการแก้ไขหรือตรวจสอบเอกสารนั้น ๆ ก่อนการพิมพ์ลงแบบฟอร์มจริง
2. เพื่อให้ฟอร์มที่ได้มีความเรียบร้อยและสวยงาม เพราะมีรูปแบบตัวอักษรให้เลือกมากมายและสามารถกรอกข้อมูลลงในแบบฟอร์มให้มีความใกล้เคียงกับฟอร์มมากที่สุด
3. เพื่อช่วยลดความผิดพลาดของข้อมูลหรือเอกสาร(ฟอร์ม) เกี่ยวกับการคำนวณ, การหาผลรวมของตัวเลข เช่นเอกสารใบส่งของ เพราะจะมีสูตรที่ช่วยในการคำนวณ และเมื่อคำนวณเรียบร้อยแล้วก็จะนำค่าที่ได้ไปลงฟิลด์ที่กำหนดไว้โดยอัตโนมัติด้วย
4. เพื่อให้งานที่ได้มีความรวดเร็วมากขึ้นเพราะสามารถนำข้อมูลเก่ามาทำการแก้ไขได้และสำหรับข้อมูลใหม่ที่ไม่มีความคล้ายคลึงกับข้อมูลเก่าเราก็สามารถกรอกข้อมูลได้รวดเร็วเพราะการย้าย CURSOR ไปตำแหน่งฟิลด์ถัดไปได้ง่ายโดยโดยไม่ต้องไปกระชาก CURSOR เอง หรือในงานที่มีฐานข้อมูลอยู่แล้วก็สามารถดึงข้อมูลจากฐานข้อมูลมาได้เลย โดยไม่ต้องมากรอกแบบฟอร์มใหม่
5. แก้ปัญหา DRIVER ของเครื่องพิมพ์เพราะโปรแกรมที่ใช้ทำงานภายใต้ WINDOWS ซึ่งตัว WINDOWS นี้จะจัดการเรื่องเครื่องพิมพ์เอง รวมทั้งเครื่องพิมพ์รุ่นใหม่ ๆ ก็จะมี DRIVER สำหรับ WINDOWS ให้มาด้วยเสมอ

บทที่ 2 ทฤษฎีและหลักการ

คอมไพเลอร์และอินเตอร์พรีเตอร์

โปรแกรมพิเศษประเภทหนึ่งชื่อว่า คอมไพเลอร์ (compilers) จะสามารถช่วยให้หลีกเลี่ยงรายละเอียดที่ยุงยากของการเขียนโปรแกรมด้วยภาษาแอสเซมบลี คอมไพเลอร์จะทำให้นักเขียนโปรแกรมสามารถเขียนคำสั่งได้อย่างมีประสิทธิภาพมากกว่าแอสเซมบลีหรือคำสั่งเครื่องเป็นอันมาก เช่น คำสั่งเบสิกหนึ่งคำสั่งมักจะแปลไปเป็นคำสั่งเครื่องจำนวนมาก นักเขียนโปรแกรมจะเขียนคำสั่งด้วยภาษาด้านฉบับ (source language) ของคอมไพเลอร์ ซึ่งหลังจากนั้นจะแปลภาษาด้านฉบับไปเป็นภาษาเครื่อง ภาษาด้านฉบับที่นิยมใช้ในปัจจุบันคือ C ปาสคาล และเบสิก รูปที่ 2.1 แสดงแนวทางการทำงานของคอมไพเลอร์

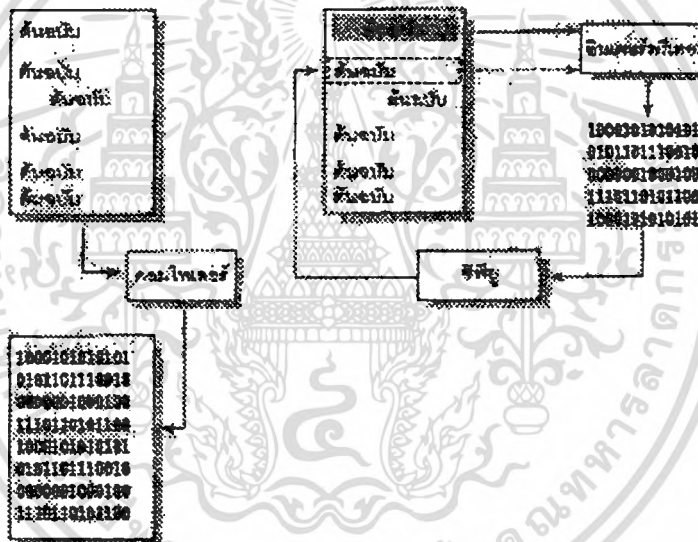
ถ้าดูอย่างง่าย ๆ แล้ว คอมไพเลอร์หรือแอสเซมเบลอร์จะอ่านไฟล์ข้อความสั่งที่นักเขียนโปรแกรมเขียนขึ้น และสร้างไฟล์รหัสเครื่องเป็นผลส่งออก หลังจากนั้นเอ็กซีคิวต์โปรแกรมรหัสเครื่องเพื่อสร้างผลลัพธ์ตามต้องการ ในระบบปฏิบัติการ MS-DOS และ Microsoft Windows ไฟล์ที่มีส่วนขยาย EXE หรือ COM จะประกอบไปด้วยรหัสเครื่อง และสามารถบรรจุเข้าไปในหน่วยความจำ และถูกเอ็กซีคิวต์ได้โดยตรง



รูปที่ 2.1

แปลภาษาด้านฉบับไปเป็นภาษาเครื่อง

อินเตอร์พรีเตอร์ (interpreters) ซึ่งแปลข้อความสั่งรหัสต้นฉบับเช่นกัน มีลักษณะที่คล้ายกับคอมไพเลอร์ อย่างไรก็ตาม อินเตอร์พรีเตอร์จะแปลรหัสต้นฉบับเพียงบรรทัดเดียว และหลังจากนั้นจะเอ็กซีคิวต์รหัสเครื่องที่เกิดขึ้นในทันที ถัดจากนั้นอินเตอร์พรีเตอร์ก็จะกลับไปยังรหัสต้นฉบับและลงมือแปลบรรทัดถัดไป รูปที่ 2.9 เป็นการเปรียบเทียบคอมไพเลอร์และอินเตอร์พรีเตอร์ อีกหนทางหนึ่งที่จะทำให้เข้าใจถึงความแตกต่างได้อย่างชัดเจนมากขึ้น ให้ลองนึกถึงว่าการแปลหนังสือไปเป็นภาษาต่างประเทศจะแตกต่างจากการตีความภาษาต่างประเทศในช่วงเวลาสั้น ๆ ที่ดำเนินการโดยล่ามของสหประชาชาติอย่างไรบ้าง กระบวนการแปลหนังสือจะคล้ายกับงานที่คอมไพเลอร์จะต้องทำ กล่าวคือผู้แปลจะรับ War and Peace ฉบับรัสเซียมา และเมื่อผ่านไประยะหนึ่งก็จะสร้างหนังสือใหม่ที่มีข้อความอย่างเดียวกันแต่เป็นฮูตู แต่ล่ามแปลรัสเซียเป็นฮูตูที่สหประชาชาติจะต้องจัดการกับข้อความที่พูดออกมาแต่ละข้อความโดยทันที ซึ่งจะเหมือนกับที่อินเตอร์พรีเตอร์จะต้องทำ



รูปที่ 2.2

การคอมไพล์และการอินเตอร์พรีตีง

คอมไพเลอร์และอินเตอร์พรีเตอร์ต่างมีข้อดีของตัวเอง คอมไพเลอร์จะดำเนินการแปลภาษาในครั้งเดียว และหลังจากนั้นให้ไฟล์เอ็กซีคิวทีเบิลที่สามารถดำเนินงานได้ตลอดเวลา อินเตอร์พรีเตอร์จะให้ผลการตอบสนองอย่างรวดเร็ว และสามารถช่วยคุณตรวจหาความผิดพลาด Visual Basic ประกอบไปด้วยทั้งคอมไพเลอร์และอินเตอร์พรีเตอร์

เราได้พบกับอินเทอร์พรีเตอร์ของ Visual Basic มาแล้ว เมื่อเลือกคำสั่ง Start จากเมนู Run Visual Basic จะแปลรหัสที่เขียนขึ้นและเริ่มต้นเอ็กซีกิวต์รหัสนั้น ในบทที่ 8 จะอธิบายว่าจะหยุดโปรแกรมไว้ชั่วคราวในขณะที่กำลังดำเนินงานอยู่นั้นได้อย่างไร และส่งคำสั่งไปยังอินเทอร์พรีเตอร์โดยตรงเพื่อที่จะหาและแก้ไขข้อผิดพลาดในรหัสโปรแกรม ในขณะที่กำลังดำเนินงานอินเทอร์พรีเตอร์นั้นทั้งโปรแกรม Visual Basic และแอปพลิเคชันจะต้องอยู่ในหน่วยความจำในเวลาเดียวกัน เราสามารถเลือกที่จะคอมไพล์แอปพลิเคชันทั้งโปรแกรม แอปพลิเคชันฉบับคอมไพล์สามารถดำเนินงานได้บนระบบคอมพิวเตอร์ใด ๆ ที่กำลังดำเนินงาน Windows โดยไม่จำเป็นต้องอาศัยโปรแกรม Visual Basic ในการดำเนินงาน

ตัวแปรและค่าคงตัว

กำหนดชนิดข้อมูลของ Visual Basic

ในบทที่ 2 เราได้กล่าวไว้ว่าบิต 01000001 จะแทนอักษร A ถ้าข้อมูลนั้นเป็นอักขระแอสกีหรือแทนค่า 65 ถ้าข้อมูลนั้นเป็นจำนวนเต็มฐานสองขนาด 8 บิต เราอาจแก้ไขความคลุมเครือเช่นนั้นได้ด้วยการพัฒนาการเข้ารหัสบิต (bit encoding) พิเศษเพื่อใช้กับสารสนเทศทุก ๆ ประเภท แต่ในกรณีของ Visual Basic ไม่ต้องทำในสิ่งนี้ ทั้งนี้เพราะ Visual Basic นิยามการแทนข้อมูลมาตรฐานเอาไว้ชุดหนึ่งเรียกว่า ชนิดข้อมูล (data types) รูปที่ 3.1 ได้ลงรายการชื่อและอักขระของชนิดข้อมูลทั้งหมดที่ปรากฏอยู่ใน Visual Basic

ชนิด integer และชนิด long เป็นจำนวนเต็มแบบ 16 บิต และ 32 บิต ง่าย ๆ ที่แทนด้วยค่าฐานสองคล้ายกับค่าที่ได้กล่าวไว้ในบทที่ 2 ทั้งสองชนิดนี้แทนเฉพาะจำนวนที่ไม่มีส่วนประกอบที่เป็นเศษส่วน integer ต้องการหน่วยความจำเพียง 16 บิต แต่มีพิสัยที่จำกัดมากกว่าชนิด long สตริง (string) เป็นอนุกรมของอักขระ ซึ่งแทนแต่ละสตริงด้วยแบบแผนการเข้ารหัสแอสกีอนุกรมที่ไม่มีอักขระบรรจบอยู่เลยเรียกว่าเป็นสตริงว่าง (empty string)

ชื่อของชนิดข้อมูล	พิสัยของค่า	หน่วยเก็บที่ต้องใช้
integer	-32768 จนถึง 32767	16บิต
Long	-21744836 จนถึง 21744836	32 บิต
String	อักขระ 0 จนถึง 65535 ตัว	8 บิตต่อ 1 อักขระ
Currency	-922377203477.0508 จนถึง 922377203477	64 บิต
Single	+1.40*10ยกกำลัง45 จนถึง 3.40*10ยกกำลัง38	32 บิต
Double	+4.94*10ยกกำลัง-324จนถึง +1.79*10ยกกำลัง38	64 บิต
Variant	พิสัยใด ๆ ที่กล่าวข้างต้นขึ้นอยู่กับค่าที่เก็บไว้	

ตารางที่ 2.1

ชนิดข้อมูลของ Visual Basic

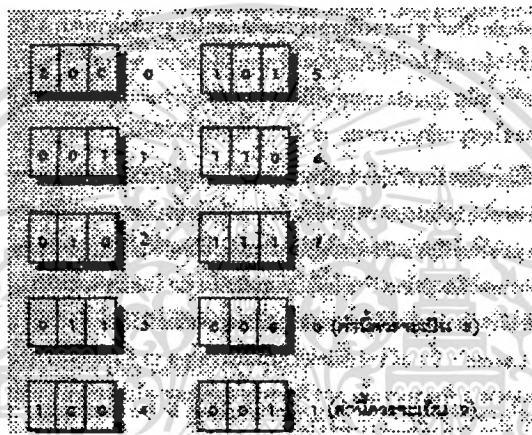
ชนิดของข้อมูล Currency ได้รับการออกแบบมาอย่างพิเศษเพื่อใช้แทนค่าเงินตรา ค่า Currency จะมีค่าความเที่ยง (precision) ของทศนิยมสี่ตำแหน่งเสมอ กล่าวคือสามารถแทนค่า 11.1625 แต่ไม่สามารถแทนค่า 21.00003 (ค่าหลังนี้จะถูกปัดให้เป็น 21.0000)

ชนิด single และ double ให้สามารถแสดงจำนวนจุดลอยตัว จำนวนจุดลอยตัว (floating point numbers) (ซึ่งบางที่เรียกว่า จำนวนจริง) แทนค่าที่มีส่วนประกอบที่เป็นเศษส่วน ชื่อจุดลอยตัวนั้นมาจากจุดทศนิยม "ลอยตัว" ซึ่งหมายถึงว่าสามารถปรากฏขึ้นที่ใดก็ได้ในจำนวนเช่นใน .000001, 356.876 และ 100000.0 ชนิดข้อมูลทั้งสองชนิดนี้แตกต่างจากชนิด currency ซึ่งชนิดหลังนี้เป็นตัวอย่างของชนิดข้อมูลจุดตายตัว Visual Basic ยังได้มีการใช้ชนิด double ในโอกาสพิเศษ ซึ่งเป็นการเข้ารหัสวัน เดือน ปี และเวลา รายละเอียดเพิ่มเติมเกี่ยวกับเทคนิคการเข้ารหัสนี้ปรากฏอยู่ในคู่มืออ้างอิงของ Visual Basic

ชนิดสุดท้ายที่กล่าวถึงคือชนิด variant ซึ่งเป็นชนิดข้อมูลประเภทที่เปลี่ยนแปลงได้ สามารถครอบครองข้อมูลค่าใด ๆ ก็ได้ ชนิด variant ประกอบไปด้วยสารสนเทศสองชั้นคือ ค่า ซึ่งอาจเป็นชนิดข้อมูลชนิดใดก็ได้ตามที่ได้บรรยายไว้ในข้างต้น และรหัสที่แสดงชนิดของค่า (เช่น currency หรือ string)

พิสัย (range)

ลักษณะเฉพาะที่สำคัญที่สุดของชนิดข้อมูลเชิงตัวเลขก็คือ พิสัยของชนิด พิสัยจะเป็นตัวจำกัดค่าที่ชนิดข้อมูลหนึ่งจะสามารถครอบครองได้ การจำกัดนี้จะขึ้นอยู่กับจำนวนการบิตที่ใช้เข้ารหัสค่า เช่น ลองพิจารณารูปที่ 3.2 ซึ่งแสดงให้เห็นว่ามีอะไรเกิดขึ้นเมื่อพยายามนับตั้งแต่ 0 ถึง 9 โดยการใช้จำนวนฐานสองขนาด 3 บิต การใช้ขนาด 3 บิตทำให้สามารถนิยามค่าได้เพียงแปดค่า ด้วยเหตุนี้สารสนเทศบางส่วนจึงสูญหายไป ซึ่งเป็นสถานะที่เรียกว่าเป็นโอเวอร์โฟลว์ (overflow) สถานะเช่นนี้เกิดขึ้นจากการที่จำนวนของบิตที่ต้องใช้ในการเก็บสารสนเทศโอเวอร์โฟลว์จำนวนของบิตที่มีอยู่



รูปที่ 2.3

การสูญเสียสารสนเทศที่เป็นผลลัพธ์ของโอเวอร์โฟลว์

ในฐานะของนักเขียนโปรแกรมจำเป็นต้องตัดสินใจว่าจะใช้ข้อมูลใดสำหรับค่าที่กำหนดให้หนึ่ง ๆ ถ้ากำลังเขียนโปรแกรมที่ง่ายมาก ๆ อาจมีเหตุผลพอสมควรที่จะหลีกเลี่ยงปัญหานี้ได้ ด้วยการใช้ชนิดข้อมูล variant เสมอ ๆ ซึ่งสามารถแทนค่าใด ๆ ก็ได้ อย่างไรก็ตาม เมื่อโปรแกรมเริ่มมีความซับซ้อนมากขึ้น จำเป็นต้องเลือกระมัดระวังมากขึ้น เพื่อจัดการกับเนื้อที่และประสิทธิภาพ (variant ไม่ค่อยมีประสิทธิภาพเพราะ Visual Basic ต้องคอยตรวจสอบค่าแต่ละค่าอย่างต่อเนื่องเพื่อคว่าเป็นชนิดข้อมูลใด)

การสร้างตัวแปรและค่าคงตัว

การเพียงแต่เก็บเข้าไว้ในหน่วยความจำไม่ว่าจะเป็นชนิดใดก็ตาม จะเป็นสิ่งที่ไร้ประโยชน์ถ้าไม่สามารถอ้างถึงข้อมูลเหล่านั้นด้วยแนวทางบางอย่าง Visual Basic ให้แนวทางที่สามารถกำหนดชื่อให้แก่หน่วยความจำเพื่อที่จะสามารถเข้าถึงข้อมูลที่เก็บอยู่ตรงตำแหน่งนั้นได้ ตำแหน่งหน่วยความจำที่กำหนดชื่อไว้นี้ เรียกว่า ตัวแปร ข้อความสั่ง Dim ซึ่งแจ้ง Visual Basic ให้ทราบถึงความตั้งใจที่จะสงวนหน่วยเก็บมีวากยสัมพันธ์หรือรูปแบบดังต่อไปนี้

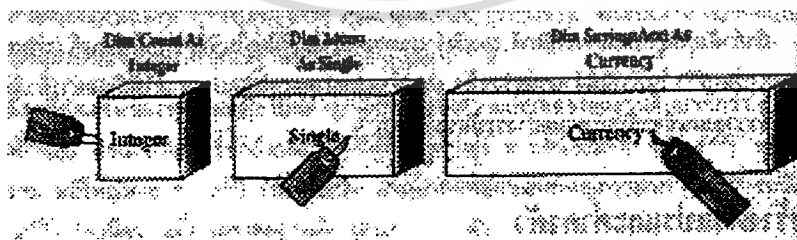
Dim variable [As Type]

เมื่อใช้ข้อความสั่งนี้ ต้องพิมพ์คำหลัก Dim และตามด้วยชื่อของตัวแปรที่ต้องการใช้ วลีที่แสดงอยู่ในวงเล็บเป็นวลีที่ใส่หรือไม่ใส่ก็ได้ โดยถ้าจะใช้วลีนี้ให้พิมพ์คำหลัก As และหลังจากนั้นระบุชนิดโดยไม่ต้องใส่วงเล็บเอาไว้ เช่น รหัสบรรทัดแรกต่อไปนี้จะบ่งบอก Visual Basic ให้สงวนเนื้อที่หน่วยความจำให้มากเพียงพอสำหรับข้อมูลชนิด currency และอ้างถึงตำแหน่งหน่วยความจำนั้นด้วยชื่อ SavingAcct

Dim SavingsAcct As Currency

การใช้ชื่อตัวแปรเช่นนี้ทำให้สามารถสั่ง Visual Basic ให้เก็บข้อมูลไว้ที่ตำแหน่งนั้น ให้ค้นคืนข้อมูล ให้คัดแปลงข้อมูล และอื่น ๆ อีกหลายอย่างได้ในภายหลัง ถ้าไม่รวมเอาวลี As เข้าไว้ด้วย Visual Basic จะถือว่าตัวแปรนั้นเป็นชนิด variant

เราอาจเปรียบเทียบตัวแปรให้เหมือนกับเป็นภาชนะหรือบ็อกซ์ข้อความสั่ง Dim จะสร้างบ็อกซ์ใหม่ขึ้นมาใบหนึ่งที่มีขนาดที่แน่นอน และกำหนดชื่อให้แก่บ็อกซ์นั้นตามที่แสดงไว้ในรูปที่ 3.3 เมื่อนำชื่อของตัวแปรเข้าไปในโปรแกรม Visual Basic ก็จะทำให้สิ่งที่บรรจุอยู่ในภาชนะนั้น (ซึ่งคือค่าล่าสุดที่เก็บไว้ ณ ที่นั้น) โปรแกรมที่เขียนในบทที่ 1 ใช้ตัวแปรสามตัวคือ StartTime, EndTime และ ElapsedTime ในขณะที่เอ็กซีคิวต์โปรแกรม โปรแกรมจะเก็บค่าข้อมูลที่แทนเวลาล่าสุด และเวลาที่ล่วงเลยไปในตัวแปรเหล่านั้น



รูปที่ 2.4

การสงวนหน่วยเก็บโดยอาศัยการประกาศตัวแปร

เมื่อสว่นเนื้อที่หน่วยความจำเพื่อใช้กับตัวแปรเชิงตัวเลข Visual Basic ก็จะกำหนดตัวแปรให้มีค่าเริ่มต้นเป็น 0 แต่การพึ่งพิงการกำหนดค่าเริ่มต้นแบบอัตโนมัติไม่ได้เป็นความคิดที่ดีนักถ้าต้องการที่จะให้ตัวแปรครอบครองค่าใดค่าหนึ่งนั้น ก็ควรจะกำหนดค่าให้แก่ตัวแปรในโปรแกรม การกำหนดค่าให้แก่ตัวแปรอย่างชัดเจนจะทำให้สามารถอ่านโปรแกรมได้ง่ายขึ้น นอกจากนี้ควรจดจำไว้ด้วยว่า การกำหนดค่าเริ่มต้นแบบอัตโนมัติเป็นลักษณะเฉพาะของ Visual Basic ถ้าแปลงโปรแกรมของคุณไปใช้กับสภาพแวดล้อมการเขียนโปรแกรมอื่น ๆ ระบบใหม่ที่ใช้อาจไม่ได้สนับสนุนการกำหนดค่าเริ่มต้น และโปรแกรมก็ไม่สามารถใช้งานได้อีกต่อไปในบาง ครั้งอาจต้องการใช้ค่าที่ไม่มีการเปลี่ยนแปลงเช่นค่า pi จำนวนอาโวกาโดร หรือจำนวนกรัมในหนึ่งออนซ์ เป็นต้น แม้ว่าสามารถใช้ค่าตัวเลขเหล่านี้ในโปรแกรมได้โดยตรง แต่การใช้ชื่อ PI แทนที่จะต้องใช้จำนวน 3.1415926 ซ้ำ ๆ กันในแต่ละครั้งที่ต้องการคุณจะเป็นการง่ายกว่ามาก

ในกรณีของค่าที่ตายตัวเช่นนั้น Visual Basic สามารถประกาศค่าคงตัวได้ ซึ่งจะสว่นตำแหน่งหน่วยความจำโดยที่เนื้อหาหรือคอนเทนตั้นจะไม่มีการเปลี่ยนแปลงเลข (ซึ่งแตกต่างไปจากเนื้อหาของตัวแปร) ถ้าพยายามจะดัดแปลงค่าคงตัว Visual Basic จะก่อให้เกิดข้อความระบุความผิดพลาด ข้อความสั่ง Const จะประกาศค่าคงตัว และมีรูปแบบดังนี้

Const name = expression

เราไม่จำเป็นต้องประกาศชนิดของค่าคงตัว เพราะ Visual Basic จะกำหนดชนิดให้ตามค่าของค่าคงตัวนั้น เช่น ถ้าเขียน Const ANSWER = 42 Visual Basic ใช้ชนิด integer หรือถ้าป้อน Const FIRSTNAME = "Ross" Visual Basic จะสร้างค่าคงตัวชนิด String ขึ้น (ชื่อของค่าคงตัวนั้นมักจะนิยมเขียนด้วยตัวอักษรตัวใหญ่ทั้งหมด)

มีบ่อยครั้งที่จะปรากฏความคลุมเครือเกี่ยวกับว่า Visual Basic ควรจะสร้างค่าคงตัวชนิดใดขึ้นมา เช่น ค่า 3.01 อาจเป็นได้ทั้งชนิด single, double และ currency แต่โดยปริยายแล้ว Visual Basic จะเลือกการแทนที่ต้องการเนื้อที่หน่วยความจำน้อยที่สุด อย่างไรก็ตาม สามารถออกคำสั่งได้ด้วยการวางอักขระการประกาศชนิดแบบพิเศษเอาไว้หลังชื่อ ให้ใช้อักขระนี้เฉพาะในการประกาศเท่านั้น ไม่ใช่ในตัวของโปรแกรม อักขระประกาศชนิดของ Visual Basic ได้นำมาลงไว้ในตารางที่ 2.2

อักขระการประกาศ	ชนิดข้อมูลที่กำหนด
%	integer
!	single (จำนวนจริง)
#	double (จำนวนจริง)
@	currency (จุดตายตัว)
\$	string (ค่าแรกเดอร์)

ตารางที่ 2.2

อักขระประกาศชนิดของ Visual Basic

ดังนั้นการประกาศ Const ONE& = 1 สงวนหน่วยความจำ 4 ไบต์สำหรับค่าคงตัวชนิด long และการประกาศ Const ONE? = 1 สงวนหน่วยความจำ 8 ไบต์ และเก็บค่าเอาไว้ในการแทนจุดลอยตัว ในทั้งสองกรณีนี้อาจใช้ชื่อ ONE (โดยไม่ต้องมีอักขระประกาศชนิด) ในโปรแกรมเพื่ออ้างถึงค่าเหล่านั้น ควรจะฝึกที่จะนิยามค่าคงตัวสำหรับค่ามาตรฐาน การทำเช่นนี้จะทำให้โปรแกรมอ่านได้ง่ายขึ้นเป็นอย่างมาก

ข้อความสั่ง นิพจน์ และกระบวนการ (Assignment)

ข้อความสั่งกำหนดค่า

ข้อความสั่งที่ง่ายที่สุดและอาจเป็นข้อความสั่งที่พบบ่อยที่สุดในโปรแกรมหนึ่ง ๆ คือ ข้อความสั่งกำหนดค่า ข้อความสั่งนี้ประกอบไปด้วยชื่อตัวแปร ตามด้วยโอเปอเรเตอร์กำหนดค่า (=) แล้วตามด้วยนิพจน์บางประเภท ในระดับที่พื้นฐานที่สุด นิพจน์อาจประกอบไปด้วยค่าคงตัวง่าย ๆ เพียงค่าเดียว อย่างไรก็ตาม นิพจน์อาจประกอบไปด้วยการอ้างอิงไปยังตัวแปรอื่น ๆ หรือการผสมกันของตัวแปรและค่าคงตัว ซึ่งอาจมีโอเปอเรเตอร์ชั้นบางอย่างอยู่ด้วย ข้อความสั่งทั้งหมดต่อไปนี้เป็นข้อความสั่งกำหนดค่าที่ถูกต้อง

```
StartTime = Now
Explorer.Name = "Captain Spaulding"
BitCount = ByteCount * 8
Energy = Mass * LIGHTSPEED 2
NetWorth = TotalAssets - TotalLiabilities
```

ข้อความสั่งกำหนดค่าจะเก็บสารสนเทศไว้ ค่าของนิพจน์ซึ่งปรากฏอยู่ทางด้านขวาของโอเปอเรเตอร์กำหนดค่าจะถูกคำนวณ และนำผลลัพธ์ที่ได้ไปเก็บไว้ในตัวแปรที่อยู่ทางด้านซ้ายของโอเปอเรเตอร์ ชนิดข้อมูลของตัวแปรจะต้องเหมาะสมหรือตรงกันกับค่าของนิพจน์ที่ถูกคำนวณไว้ เช่น เราไม่สามารถเก็บค่าคงตัวหรือนิพจน์สตริงไว้ในตัวแปรชนิด integer หรือในตัวแปรชนิด double ถ้าชนิดข้อมูลเป็นชนิดที่เกี่ยวข้องกัน แต่ไม่เหมือนกันทีเดียวนัก เช่น อาจต้องการเก็บค่าจำนวนเต็มไว้ในตัวแปรชนิดจุดลอยตัว Visual Basic จะแปลงผลลัพธ์ของนิพจน์ไปเป็นชนิดของตัวแปรนั้น อย่างไรก็ตาม ถ้าเก็บผลลัพธ์ของนิพจน์ไว้ในตัวแปรชนิด variant ก็ยังคงชนิดของนิพจน์ไว้ ตัวแปรชนิด variant จะบันทึกทั้งชนิดและค่าไว้

โชคไม่ดีนัก โอเปอเรเตอร์กำหนดค่าจะเหมือนกันกับสัญลักษณ์ทางคณิตศาสตร์ที่ใช้แทนความเท่ากัน (เครื่องหมายเท่ากัน เช่นในสมการ $2+1=1+2$) ความเท่ากันในนัยคณิตศาสตร์แสดงสถานะที่เป็นจริง ในข้อความสั่งกำหนดค่าของ Visual Basic สัญลักษณ์ = เป็นคำสั่งที่วางผลลัพธ์ของนิพจน์ที่ตามมาเอาไว้ในตัวแปรที่อยู่หน้า สัญลักษณ์นี้แทนการทำงานที่ระบบต้องทำมากกว่าจะเป็นข้อความสั่งของข้อเท็จจริง สมการ $2+1=1+2$ เป็นข้อความสั่งที่ไม่ถูกต้องใน Visual Basic ทั้งนี้เพราะไม่มีตัวแปรอยู่ทางด้านซ้ายของเครื่องหมายเท่ากับ ถ้าพยายามป้อนรหัสบรรทัดนั้น Visual Basic จะรายงานความผิดพลาดออกมา

ข้อความสั่งโปรแกรมทั้งหมดที่อยู่ในแอปพลิเคชัน Stopwatch ที่สร้างขึ้นเป็นข้อความสั่งกำหนดค่า ในบางข้อความสั่งเช่น `btnStop.Enabled = True` เป็นการกำหนดค่าของค่าคงตัวง่าย ๆ ในขณะที่บางข้อความสั่งเช่นข้อความสั่ง `ElapsedTime = EndTime-StartTime` เป็นการใช้นิพจน์ ในข้อความสั่งเช่น `txtStart.Txt = Format(StartTime, "hh:mm:ss")` เป็นการเรียกฟังก์ชันแบบในตัว (built-in function) ส่วนที่เหลือของบทนี้จะสำรวจความสามารถของ Visual Basic ในสาขาเหล่านี้

นิพจน์และโอเปอเรเตอร์

ตามที่เราได้กล่าวไว้ก่อนหน้านี้แล้วว่านิพจน์อาจเป็นเพียงค่าคงตัวค่าหนึ่ง แต่โดยทั่วไปนิพจน์มักจะแทนการคำนวณที่ต้องทำ ตามปกติโอเปอเรเตอร์จะบ่งบอกชนิดของการคำนวณ เราอาจจะคุ้นเคยกับโอเปอเรเตอร์คำนวณ เช่น เครื่องหมายบวก (+) ซึ่งแสดงถึงการบวก นิพจน์ $2+1$ บ่งบอก Visual Basic ให้บวกค่า 2 และ 1 เข้าด้วยกัน 2 และ 1 นั้นเป็นโอเปอเรนด์ของการบวก นอกเหนือจากโอเปอเรเตอร์คำนวณแล้ว Visual Basic ยังประกอบไปด้วยโอเปอเรเตอร์เชื่อมต่อสตริง โอเปอเรเตอร์เปรียบเทียบ และโอเปอเรเตอร์เชิงตรรกะ

โดยตัวมันเองนั้น นิพจน์เป็นข้อความสั่งที่ใช้ไม่ได้หรือไม่ถูกต้องเพราะตัวของนิพจน์ไม่ได้มีความสมบูรณ์ เช่น การเพียงแต่ป้อนบรรทัด 2+1 เข้าไปในโปรแกรม Visual Basic ทำให้เกิดความผิดพลาดขึ้น ถ้าเปรียบข้อความสั่งในภาษา Visual Basic เสมือนกับเป็นประโยค นิพจน์จะเปรียบเสมือนกับวลี เราสามารถวางนิพจน์ไว้ตรงที่ใดก็ได้ที่สามารถใช้ค่าของค่าคงตัวการนิยามนิพจน์ไม่มีเทคนิคที่ตายตัว การนิยามบางส่วนอาจประกอบไปด้วยกฎเกณฑ์เหล่านี้

* ค่าคงตัวเป็นนิพจน์

* การรวมกันของนิพจน์แล้วตามด้วยโอเปอเรเตอร์ แล้วตามด้วยนิพจน์อื่น ๆ เป็นนิพจน์เช่นกันตามกฎเหล่านี้ 3 เป็นนิพจน์เช่นเดียวกันกับ 1 นอกจากนี้ $3+1$ เป็นนิพจน์เช่นกัน และที่ตามมาคือ $3*3+1$ เป็นนิพจน์ด้วยเช่นเดียวกันกับ $3*3+1+1$ จากลักษณะเช่นนี้จะเห็นได้ว่านิพจน์สามารถซับซ้อนได้อย่างไม่จำกัด

ชนิดของนิพจน์

ชนิดของนิพจน์จะถูกกำหนดไว้ด้วยชนิดของโอเปอเรนด์ สำหรับค่าของค่าคงตัวหนึ่ง ชนิดของนิพจน์จะเหมือนกันกับชนิดของค่าคงตัว สำหรับนิพจน์ที่ประกอบไปด้วยโอเปอเรชัน Visual Basic ใช้ชนิดของโอเปอเรนด์ทั้งสองตัวเพื่อหาชนิดของผลลัพธ์ ตามปกติถ้าบวกจำนวนเต็มสองจำนวน ผลลัพธ์จะเป็นจำนวนเต็ม ถ้าบวกค่าชนิด double สองค่า ผลลัพธ์จะเป็นชนิด double ถ้าใช้ชนิดต่าง ๆ ผสมกัน เช่น การคูณจำนวนเต็มด้วยชนิด single เช่นที่ปรากฏอยู่ในนิพจน์ $2*7.5$ โดยทั่วไป Visual Basic จะทำผลลัพธ์ให้เป็นชนิดที่มีพิสัยที่กว้างที่สุด ซึ่งในกรณีนี้คือจำนวนจริงชนิด-single อย่างไรก็ตาม เพื่อไม่ให้เกิดความคลุมเครือ จึงไม่ควรใช้หลาย ๆ ชนิดผสมกัน

เราอาจใช้อักขระการประกาศชนิดกับค่าคงตัวเชิงตัวเลข เพื่อช่วยหลีกเลี่ยงการใช้ชนิดหลาย ๆ ชนิดผสมกัน อักขระเหล่านี้จะผลักดันให้ค่าคงตัวครอบครองชนิดเฉพาะชนิดใดชนิดหนึ่ง เช่น ตามปกติ 100 จะเป็นจำนวนเต็ม แต่ 100! จะเป็นชนิด single และ 100@ เป็นค่าชนิด currency (อักขระการประกาศชนิดของ Visual Basic ปรากฏอยู่ในบทที่ 3 รูปที่ 3.4)

โอเปอเรเตอร์คำนวณ

โอเปอเรเตอร์ที่คุ้นเคยกันมากที่สุดได้แก่ โอเปอเรเตอร์ที่ทำการคำนวณอย่างง่าย ๆ ตารางที่ 2.3 ได้ลงรายการโอเปอเรเตอร์คำนวณที่มีอยู่ใน Visual Basic

ในการบวก การลบ และการคูณ ผลลัพธ์ที่ได้เป็นจำนวนที่มีชนิดเดียวกันกับโอเปอเรนด์ (ชนิดผสมได้กล่าวเอาไว้ในตอนก่อนหน้านี้แล้ว) โดยทั่วไปการใช้จำนวนเต็มหรือจำนวนจุดลอยตัวจะเร็วกว่าการใช้จำนวนจริง แต่อาจสังเกตไม่พบความแตกต่างเว้นแต่ว่าทำการคำนวณที่ซับซ้อนมาก ๆ

การบวก การลบ และการคูณจะประพฤติตามที่คาดหมายไว้ เราต้องใช้โอเปอเรเตอร์ยกกำลัง เพราะไม่สามารถใช้ตัวเลขซูเปอร์สคริปต์ใน Visual Basic และโอเปอเรเตอร์สามตัวที่เหลืออยู่ที่เกี่ยวข้องกับการหารเป็นโอเปอเรเตอร์ที่ซับซ้อนขึ้นเล็กน้อย

โอเปอเรเตอร์	โอเปอเรชันที่ทำ
+	การบวก
-	การลบ
*	การคูณ
^	การยกกำลัง
/	การหาจุดลอยตัว
\	การหารจำนวนเต็ม
Mod	มอดุลัส

ตารางที่ 2.3

โอเปอเรเตอร์คำนวณ

การยกกำลัง

โอเปอเรเตอร์ยกกำลังใช้ในการคำนวณกำลังและราก ในคณิตศาสตร์จะแสดงการยกกำลังโดยใช้ซูเปอร์สคริปต์ เช่น ค่า 2 ยกกำลัง 8 ซึ่งใน Visual Basic จะคำนวณด้วยนิพจน์ 2⁸ รากที่สองของ 2 ซึ่งคือ 2 ยกกำลัง 1/2 อาจคำนวณโดยใช้นิพจน์ 2^(1/2) หรือนิพจน์ 20.5 ตารางที่ 2.4 เป็นตัวอย่างของการยกกำลัง

2.4.10

นิพจน์	ผลลัพธ์	ค่าที่คำนวณ
10^2	100	10×10
10^3	1000	$10 \times 10 \times 10$
10^{-2}	0.01	ค่าผกผันของ 10 ยกกำลัง 2 หรือ $1/100$
$25^{0.5}$	5	รากที่สองของ 25 หรือ 25 ยกกำลัง $1/2$
$8^{(1/3)}$	2	รากที่สามของ 8 หรือ 8 ยกกำลัง $1/3$

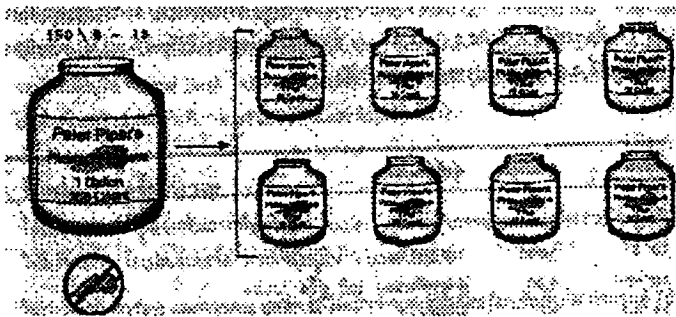
ตารางที่ 2.4

ตัวอย่างของการยกกำลัง

การหารจำนวนเต็มและการหารจำนวนจุดลอยตัว

โอเปอเรเตอร์หารจุดลอยตัว (/) ทำการหารแบบมาตรฐาน แต่ส่งคืนผลลัพธ์ที่เป็นจำนวนจุดลอยตัว เช่น นิพจน์ $3/2$ ประเมินค่าได้เป็น 1.5 ตามที่คาดหมายไว้ ส่วนโอเปอเรเตอร์หารจำนวนเต็ม นั้น (\) ส่งคืนผลลัพธ์ที่เป็นค่าจำนวนเต็ม เมื่อใช้โอเปอเรเตอร์นี้ นิพจน์ $3 \setminus 2$ ประเมินค่าได้เป็น 1 ลักษณะเช่นนี้อาจทำให้คิดว่าฟังก์ชันคำนวณส่งคืนคำตอบผิด ๆ

การที่จะคิดเช่นได้นั้นขึ้นอยู่กับทฤษฎี ลองดูตัวอย่างต่อไปนี้ สมมติว่าคุณเป็นผู้ที่บรรจุหีบห่อพริกทอง และต้องการทราบว่าสามารถบรรจุพริกทองเป็นจำนวนเท่าใดเข้าไปในไหขนาด 1 ไพท (เท่ากับ $1/8$ แกลลอน) ในฐานะของผู้เชี่ยวชาญสาขานี้ คุณทราบว่าสามารถบรรจุพริกทอง 150 ชิ้นในไหขนาด 1 แกลลอน และคุณไม่เคยบรรจุพริกทองที่ไม่เต็มชิ้น ไหขนาด 1 แกลลอนนี้มีขนาดใหญ่กว่าไหขนาด 1 ไพทอยู่ 8 เท่า ดังนั้นคุณสามารถบรรจุพริกทองได้ $150 \setminus 8$ นั่นคือ 18 ชิ้นเข้าไปในไหขนาด 1 ไพทแต่ละใบ ในกรณีนี้คุณต้องใช้โอเปอเรเตอร์หารจำนวนเต็มในการคำนวณเช่นที่แสดงอยู่ในรูปที่ 2.5 ทั้งนี้เพราะคุณต้องการบรรจุเฉพาะพริกทองที่เป็นชิ้นเต็ม ๆ เท่านั้น การทำการคำนวณโดยอาศัยโอเปอเรเตอร์หารจุดลอยตัว ($150/8$) จะให้ผลลัพธ์เป็น 18.75 ซึ่งเป็นค่าที่ไม่เหมาะสมในสถานะการณ์เช่นนี้



รูปที่ 2.5

การใช้โอเปอเรเตอร์หารจำนวนเต็ม

เนื่องจาก Visual Basic สามารถแปลงชนิดข้อมูลระหว่างกันได้ จึงอาจคิดว่าอาจจะทำให้ได้ผลลัพธ์อย่างเดียวกันนี้ได้โดยเพียงแต่กำหนดผลลัพธ์ของการหารจุดลอยตัวให้แก่ตัวแปรจำนวนเต็ม ตามที่แสดงไว้ในรหัสต่อไปนี้

```
Dim PintJar As Integer
```

```
PintJar = 150/8
```

ตามรหัสข้างต้นนี้ เมื่อได้เอ็กซีคิวต์ข้อความสั่งกำหนดค่าแล้ว ค่า *PintJar* จะเป็น 19 แทนที่จะเป็น 18 โดยในขั้นแรก Visual Basic จะคำนวณนิพจน์ $150/8$ ซึ่งให้ผลลัพธ์ 18.75 เนื่องจากต้องการให้เก็บค่าไว้ในตัวแปรจำนวนเต็ม Visual Basic จึงปัดเศษจำนวนจุดลอยตัวให้เป็นจำนวนเต็มที่ใกล้ที่สุด ซึ่งในกรณีนี้คือ 19 เมื่อใช้โอเปอเรเตอร์จำนวนเต็ม ส่วนที่เป็นเศษของผลลัพธ์จะถูกตัดออกไปแทนที่จะปัดเศษ ตารางที่ 2.5 เปรียบเทียบความแตกต่างระหว่างการตัดและการปัดเศษ

ค่าแรกเริ่ม	ตัดเป็น	ปัดเศษเป็น
24.1	24	24
24.5	24	24
24.50001	24	25
24.9999	24	25

ตารางที่ 2.5

การตัดและการปัดเศษ

มอดูลัส

โอเปอเรเตอร์ Mod ซึ่งมีความเกี่ยวข้องกับการหารจำนวนเต็มอย่างใกล้ชิด จะส่งคืนส่วนที่เหลือที่เกิดขึ้นจากการหารโอเปอเรนด์ตัวแรกด้วยโอเปอเรนด์ตัวที่สอง เช่น ถ้าหาร 7 ด้วย 4 ผลลัพธ์ที่ได้คือ 1 พร้อมกับมีส่วนคงเหลือเท่ากับ 3 ดังนั้นนิพจน์ $7 \text{ Mod } 4$ ประเมินค่าได้เป็น 3 ในทำนองเดียวกัน นิพจน์ $21 \text{ Mod } 4$ ให้ผลลัพธ์เป็น 1 (21 หารด้วย 4 ได้เท่ากับ 5 พร้อมกับมีส่วนคงเหลือเท่ากับ 1)

ลองมาดูตัวอย่างพริกคองอีกครั้ง ในตอนนี้รู้แล้วว่าสามารถบรรจุพริกคองได้ 18 ชิ้นเข้าไปในไหขนาดหนึ่งไพท ถ้ามีพริกคองที่ต้องบรรจุอยู่ 75 ชิ้น นิพจน์ $75/8$ จะบอกจำนวนของไหขนาด 1 ไพทที่ต้องใช้ (4) และนิพจน์ $75 \text{ Mod } 18$ จำนวนจำนวนของพริกคองที่เหลืออยู่ (3) ดังที่แสดงในรูปที่ 2.6



รูปที่ 2.6

การใช้โอเปอเรเตอร์หารจำนวนเต็ม และโอเปอเรเตอร์ Mod

การทําก่อนของโอเปอเรเตอร์คำนวณ

ตารางที่ 2.6 แสดงลำดับชั้นการทําก่อนของโอเปอเรเตอร์คำนวณ การยกกำลังมีการทําก่อนสูงที่สุดส่วนการบวกและการลบจะอยู่ต่ำที่สุดในลำดับชั้น โอเปอเรเตอร์ที่ลงอยู่บนบรรทัดเดียวกันมีการทําก่อนที่เท่ากัน

โอเปอเรเตอร์	โอเปอเรชันที่ทำ
$\wedge$	การยกกำลัง
$*/$	การคูณ การหารจุดลอยตัว
$\backslash$	การหารจำนวนเต็ม
Mod	มอดุลัส (ส่วนคงเหลือ)
$+ -$	การบวก การลบ

ตารางที่ 2.6

โอเปอเรเตอร์คำนวณในลำดับของการทำก่อน

ตารางที่ 2.7 แสดงตัวอย่างของนิพจน์ใน Visual Basic และค่าที่เกิดขึ้นเมื่อกำหนดนิพจน์แล้ว
อย่างง่ายถึงลำดับขั้นการทำก่อนที่แสดงในรูปที่ 4.6 เพื่อที่ว่า Visual Basic ทำให้ได้ผลลัพธ์อย่างไร

นิพจน์	ผลลัพธ์	คำอธิบาย
$3+2*7$	17	การคูณมีการทำก่อนสูงกว่า
$(3+2)*7$	35	วงเล็บซ้อนทับการทำก่อน
$1+((2+3)*2)*2$	21	วงเล็บที่ซ้อนอยู่จะถูกประเมินเป็นอันดับแรก
$14\backslash 5*2$	1	การหารจุดลอยตัวและการคูณมีการทำก่อนเท่ากันโอเปอเรชันจะทำจากซ้ายไปขวา
$27/3$	9	การยกกำลังมีการทำก่อนสูงกว่า
$27(1/3)$	3	วงเล็บซ้อนทับการทำก่อน

ตารางที่ 2.7

การใช้กฎการทำก่อนเพื่อคำนวณค่า

โอเปอเรเตอร์เชื่อมต่อสตริง

เมื่อโอเปอเรนด์ในนิพจน์เป็นสตริง โอเปอเรเตอร์บวก (+) ทำหน้าที่เชื่อมต่อหรือรวมสตริงเข้าด้วยกัน เราสามารถใช้การเชื่อมต่อเพื่อรวมสตริงสองสายหรือมากกว่าเข้าด้วยกัน เพื่อสร้างเป็นสตริงที่ยาวกว่าเดิมหนึ่งสาย ผลลัพธ์ของการเชื่อมต่อเป็นสตริงใหม่ทั้งหมดคคคคที่ไม่มีการดัดแปลงโอเปอเรนด์ดั้งเดิมเลยแม้แต่หนึ่งท่อนโปรแกรมต่อไปนี้ใช้การเชื่อมต่อเพื่อก่อกำเนิดบรรทัดจาก Lewis Carroll

```
x$ = "bats"
```

```
y$ = "cats"
```

```
Debug.Print "Do" + x$ + "eat" + y$ + "?" 'Do bats eat cats?
```

```
Debug.Print "Do" + y$ + "eat" + x$ + "?" 'Do cats eat bats?
```

ขอให้จำไว้ว่าในกรณีนี้โอเปอเรนด์จะต้องเป็นสตริงเสมอ ถ้าต้องการเชื่อมต่อจำนวนเข้ากับสตริงเพื่อจุดประสงค์เกี่ยวกับการแสดงผล จะต้องใช้ฟังก์ชันแบบในตัวของ Visual Basic ฟังก์ชันหนึ่งเพื่อแปลงจำนวนไปเป็นสตริง ฟังก์ชันดังกล่าวที่ง่ายที่สุดคือ Str\$ เช่น ในนิพจน์ "The square root of 2 is" + Str\$(2^0.5) ฟังก์ชันนี้จะแปลงจำนวนที่อยู่ในวงเล็บไปเป็นสตริงควรจะรวมเอาช่องว่างเข้าไปในสตริงด้วยเวลาจำเป็น มิฉะนั้นแล้ว Visual Basic จะเพียงแต่รวมสตริงที่ให้ไว้เท่านั้นโดยไม่มีช่องว่าง

โอเปอเรเตอร์เปรียบเทียบ

ตารางที่ 2.8 ได้ลงรายการโอเปอเรเตอร์เปรียบเทียบไว้ทั้งหมดหกตัว เมื่อเปรียบเทียบค่าสองค่าผลลัพธ์ที่ได้จากโอเปอเรชันเป็นค่า Boolean นั่นคืออาจเป็น True หรือ False (คำว่า Boolean เป็นคำที่ได้มาจากชื่อของนักคณิตศาสตร์ชาวอังกฤษ จอร์จ บูล) Visual Basic ประกอบไปด้วยค่าคงตัวแบบในตัวชื่อว่า True และ False ที่แทนค่าจำนวนเต็ม -1 และ 0 ตามลำดับการแทนของ 0 เป็นจำนวนฐานสองที่ตั้งบิตทั้งหมดให้เป็น 0 และการแทนของ -1 เป็นจำนวนฐานสองซึ่งตั้งบิตทั้งหมดให้เป็น 1 ซึ่งตั้ง True ให้อยู่ตรงข้ามกับ False

โอเปอเรเตอร์	การเปรียบเทียบ
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าหรือเท่ากับ
<=	น้อยกว่าหรือเท่ากับ
=	เท่ากับ
<>	ไม่เท่ากับ

ตารางที่ 2.8

โอเปอเรเตอร์เปรียบเทียบ

ตารางที่ 2.9 ได้ลงการประเมินและคำอธิบายนิพจน์เปรียบเทียบบางนิพจน์ โอเปอเรเตอร์เปรียบเทียบเรียกได้อีกอย่างหนึ่งว่าเป็น โอเปอเรเตอร์ เชิงสัมพันธ์ (relational operators) ทั้งนี้เพราะโอเปอเรเตอร์เหล่านี้จะประเมินสัมพันธภาพของโอเปอเรนด์สองตัว แม้ว่าโอเปอเรเตอร์เปรียบเทียบมีการทำงานที่เท่ากันในหมู่ของตัวเอง แต่การทำงานของโอเปอเรเตอร์เปรียบเทียบใด ๆ จะต่ำกว่าโอเปอเรเตอร์คำนวณทุกตัว

นิพจน์	ผลลัพธ์	คำอธิบาย
3+1>3	True	4 มากกว่า 3
"abc" <> "a" + "b" + "c"	False	หลังการเชื่อมต่อสตริงทั้งสองเท่ากัน
("abc")		
"quid" <= "pro quo"	False	เปรียบเทียบสตริงโดยเรียงตามลำดับอักษร
ไม่ได้เปรียบเทียบ		เทียบความยาว
2.3>=1.1*2	True	2.3 มากกว่า 2.2
0=(2<1)	True	2<1 เป็น False ซึ่งเท่ากับ 0

ตารางที่ 2.9

การใช้โอเปอเรเตอร์เปรียบเทียบ

ที่ผ่านมามีอาจสังเกตเห็นว่าโอเปอเรเตอร์ “เท่ากับ” ใช้สัญลักษณ์เดียวกันกับโอเปอเรเตอร์กำหนดค่า อย่างไรก็ตาม สิ่งนี้ไม่ควรสร้างความสับสนมากนัก ทั้งนี้เพราะจะใช้โอเปอเรเตอร์กำหนดค่าได้ในตำแหน่งเดียวตำแหน่งเดียว นั่นคือหลังตัวแปรในข้อความสั่งกำหนดค่า ในกรณีอื่น ๆ ทั้งหมดเครื่องหมายเท่ากับจะเป็นโอเปอเรเตอร์เปรียบเทียบหรือเชิงสัมพันธ์ เช่น ในข้อความสั่ง $Test = a = b$ จะคำนวณนิพจน์ $a = b$ และให้ True หรือ False อย่างใดอย่างหนึ่ง หลังจากนั้นกำหนดค่าที่เกิดขึ้นให้แก่ตัวแปร $Test$

โอเปอเรเตอร์ตรรกะ

ในบางครั้งเรียกโอเปอเรเตอร์ตรรกะว่าเป็นโอเปอเรเตอร์ Boolean เพราะโอเปอเรเตอร์เหล่านี้ได้มาจากพีชคณิตของตรรกะคณิตศาสตร์ ซึ่งพัฒนาขึ้นโดยจอร์จ บูล โอเปอเรเตอร์เหล่านี้มีค่า Boolean เป็นโอเปอเรนด์ และโอเปอเรเตอร์จะส่งคืนผลลัพธ์ Boolean รูปที่ 4.10 ลงรายการโอเปอเรเตอร์ตรรกะโดยเรียงลำดับตามการมาก่อน (โอเปอเรเตอร์ตรรกะมีการทำการที่ต่ำกว่าโอเปอเรเตอร์เปรียบเทียบ) โอเปอเรเตอร์ตรรกะที่ง่ายที่สุดเรียกว่า Not โอเปอเรเตอร์ตัวนี้จะอยู่นำหน้าโอเปอเรนด์ตัวเดียวและส่งคืนค่าตรรกะที่ตรงกันข้าม นั่นคือนิเสธของโอเปอเรนด์ เช่น นิพจน์ $Not\ False$ ให้ผลลัพธ์เป็น True (ตรงกันข้ามกับ False) และนิพจน์ $Not(4 < 3)$ ประเมินได้เป็น True ทั้งนี้เพราะนิพจน์ $4 < 3$ เป็น False โอเปอเรเตอร์ Not ยังมีการทำก่อนอยู่สูงที่สุดในหมู่ของโอเปอเรเตอร์ Boolean ทั้งหมด โอเปอเรเตอร์ถัดไปในอันดับการมาก่อนคือ And โอเปอเรเตอร์นี้ส่งคืนผลลัพธ์ที่เป็น True ก็เฉพาะเมื่อโอเปอเรนด์ทั้งสองตัวเป็น True ถ้าโอเปอเรนด์ตัวหนึ่งตัวใดเป็น False โอเปอเรนด์ And จะส่งคืนผลลัพธ์ที่เป็น False ลักษณะการใช้เช่นนี้จะสอดคล้องกับการใช้คำว่า and ในภาษาอังกฤษ เช่นที่แสดงในประโยค “I will be impressed only if he has a Ph.D. and if he is taller than 6 foot 2” (ฉันจะประทับใจก็เฉพาะเมื่อเขามี Ph.D. และเขาสูงกว่า 6 ฟุต 2)

นิพจน์	ผลลัพธ์
False And False	False
False And True	False
True And False	False
True And True	True



โอเปอเรเตอร์ Or ส่งคืนผลลัพธ์ที่เป็น True ถ้าโอเปอเรนด์ตัวใดตัวหนึ่งในสองตัวนั้นเป็น True หรือถ้าทั้งสองตัวเป็น True และอีกครั้งหนึ่งที่ความหมาย Boolean คล้ายคลึงกับการใช้คำในภาษาอังกฤษ เช่น “I will buy the cat if it costs less than \$50 or if it is a Siamese.” (ฉันจะซื้อแมวถ้าแมวมีราคาน้อยกว่า \$50 หรือถ้าแมวเป็นแมวสยาม)

นิพจน์	ผลลัพธ์
False or False	False
False or True	True
True or False	True
True or True	True

แม้ว่าโอเปอเรเตอร์ Not, And และ Or เป็นโอเปอเรเตอร์ที่นิยมใช้กันมากที่สุด แต่ Visual Basic ยังประกอบไปด้วยโอเปอเรเตอร์อื่น ๆ อีกสามตัวคือ Xor,Eqv และ Imp โอเปอเรเตอร์ Xor (exclusive or) ส่งคืนผลลัพธ์ที่เป็น True ถ้าโอเปอเรนด์ตัวใดตัวหนึ่งเป็น True แต่ถ้าทั้งสองตัวเป็น True จะส่งคืน False กล่าวอีกนัยหนึ่ง โอเปอเรเตอร์ชนิดนี้ทำงานคล้ายกับที่โอเปอเรเตอร์ Or ทำ ยกเว้นแต่ในกรณีที่โอเปอเรนด์ทั้งสองตัวเป็น True

นิพจน์	ผลลัพธ์
False Xor False	False
False Xor True	True
True Xor False	True
True Xor True	False

โอเปอเรเตอร์ Eqv (สมมูล) ส่งคืนผลลัพธ์ที่เป็น True เมื่อโอเปอเรเตอร์ทั้งสองตัวนั้นมีค่าเดียวกัน

นิพจน์	ผลลัพธ์
False Eqv False	True
False Eqv True	False
True Eqv False	False
True Eqv True	True

เมื่อใช้โอเปอเรเตอร์ Imp (บอกเป็นนัย) ผลลัพธ์ที่ได้จะเป็น False เฉพาะเมื่อโอเปอเรนด์ตัวแรกเป็น True และโอเปอเรนด์ตัวที่สองเป็น False ในกรณีอื่น ๆ ที่เหลือ ผลลัพธ์ เป็น True โอเปอเรเตอร์ Imp เป็นโอเปอเรเตอร์ Boolean เพียงตัวเดียวที่อันดับของโอเปอเรนด์ทำให้ได้ผลลัพธ์ที่แตกต่างกัน ในบางครั้งจะการใช้การบอกเป็นนัยกับโอเปอเรเตอร์ And เช่นที่ปรากฏอยู่ใน “Socrates is a man and all men are mortal implies that Scorates is mortal” (โซเครตีสเป็นผู้ชายคนหนึ่ง และผู้ชายทั้งหมดต้องตาย บอกเป็นนัยว่าโซเครตีสต้องตาย)

นิพจน์	ผลลัพธ์
False Imp False	True
False Imp True	True
True Imp False	False
True Imp True	True

เราอาจใช้โอเปอเรเตอร์ตรรกะมากกว่าหนึ่งตัวในนิพจน์หนึ่ง ๆ รูปที่ 4.11 เป็นตัวอย่างของการใช้โอเปอเรเตอร์ตรรกะมากกว่าหนึ่งตัว ให้ลองประเมินแต่ละนิพจน์ และหลังจากนั้นตรวจสอบผลลัพธ์เทียบกับอันดับการมาก่อนตามที่แสดงไว้ในตารางที่ 2.10

โอเปอเรเตอร์	โอเปอเรชัน
Not	(ตรรกะ) “ไม่”
And	(ตรรกะ) “และ”
Or	(ตรรกะ) “หรือ”
Xor	(ตรรกะ) “หรือ” เฉพาะ
Eqv	(ตรรกะ) “สมมูล”
Imp	(ตรรกะ) “บอกเป็นนัย”

ตารางที่ 2.10

โอเปอเรเตอร์ตรรกะหรือ Boolean เรียงลำดับตามการมาก่อน

นิพจน์	ผลลัพธ์	คำอธิบาย
Not False And True	True	เทียบเท่ากับ (Not False) And True ตามลำดับการมาก่อน
True And False Xor False	False	เทียบเท่ากับ (True And False) Xor False ตามลำดับการมาก่อน
True Eqv True Xor False	False	เทียบเท่ากับ True Eqv (True Xor True) ตามลำดับการมาก่อน

ตารางที่ 2.11

การใช้โอเปอเรเตอร์ตรรกะหลาย ๆ ตัวในนิพจน์หนึ่ง ๆ

การมาก่อน

ตารางที่ 2.12 ลงรายการของชุดของโอเปอเรเตอร์ visual Basic ทั้งหมด (ยกเว้นโอเปอเรเตอร์เชื่อมต่อสตริง) โดยเรียงลำดับตามการมาก่อนจากสูงสุดไปยังต่ำสุด เมื่อพบกับนิพจน์เชิงซ้อนหนึ่ง ๆ ต้องหาว่าโอเปอเรเตอร์ใดที่มีการมาก่อนอยู่ในลำดับที่สูงสุด แล้วประเมินนิพจน์ย่อยนั้นเป็นนิพจน์แรกหลังจากนั้นก็หาว่าโอเปอเรเตอร์ใดที่มีการมาก่อนอยู่ในลำดับรองลงมา ให้ ประเมินนิพจน์ย่อยนั้น และทำเช่นนี้ต่อเนื่องไปจนกว่าจะได้ผลลัพธ์ในขั้นสุดท้าย

โอเปอเรเตอร์	โอเปอเรชัน
$\wedge$	การขกกำลัง
$*/$	การคูณ การหารจุดลอยตัว
$\backslash$	การหารจำนวนเต็ม
Mod	มอดุลัส (ส่วนคงเหลือ)
$+ -$	การบวก การลบ
$> \diamond = < = \diamond$	เปรียบเทียบ
Not	(ตรรกะ) “ไม่”
And	(ตรรกะ) “และ”
Or	(ตรรกะ) “หรือ”
Xor	(ตรรกะ) “หรือ” เฉพาะ
Eqv	(ตรรกะ) “สมมูล”
Imp	(ตรรกะ) “บอกเป็นนัย”

ตารางที่ 2.12

โอเปอเรเตอร์ของ Visual Basic เรียงลำดับตามการทำการก่อน

โอเปอเรเตอร์บางตัว เช่น การบวกและการลบ หรือชุดของโอเปอเรเตอร์เปรียบเทียบทั้งหมดมีการทำการก่อนที่เท่ากัน เมื่อพบโอเปอเรเตอร์ที่มีการทำการก่อนที่เท่ากัน การประเมินนิพจน์ย่อให้ประเมินจากซ้ายไปขวา

ลองพิจารณานิพจน์ต่อไปนี้

```
Temperature >= 451 And Flammable = True
```

เนื่องจากโอเปอเรเตอร์เปรียบเทียบสองตัว ($\geq$ และ $=$) มีการทำการอยู่ในระดับที่เท่ากัน Visual Basic ประเมินโอเปอเรเตอร์เหล่านี้จากซ้ายไปขวา หลังจากนั้น Visual Basic จึงทำโอเปอเรชัน And ในกรณีนี้ถ้า Temperature มีค่าน้อยกว่า 415 หรือ Flammable เป็น False Visual Basic ก็จะหาได้ว่านิพจน์นั้นเป็นเท็จ

ถัดไปเป็นตัวอย่างที่ซับซ้อนมากขึ้น

```
Count * 2 < 15 Or Color = "Blue" And Count * 2 < 21
```

เมื่อดำเนินการ โดยดูตามตารางการทำก่อน ในขั้นแรก Visual Basic จะทำการคุณสมบัติก่อน แล้วหลังจากนั้นจึงเปรียบเทียบสามตัวจากซ้ายไปขวา ถัดไป Visual Basic จะทำโอเปอเรชัน And และในขั้นสุดท้ายทำโอเปอเรชัน Or

ลักษณะเช่นนี้อาจทำให้หลงได้ง่าย ดังนั้นเป็นการดีที่จะเดิมวงเล็บเข้าไปเพื่อช่วยทำให้มั่นใจว่าคุณและ Visual Basic มีความคิดสอดคล้องกันเกี่ยวกับลำดับของโอเปอเรชันในตัวอย่างข้างต้นถ้าใส่วงเล็บตามแนวทางที่ Visual Basic เห็นจะเป็นดังนี้

$((\text{Count} * 2) < 15) \text{ Or } ((\text{Color} = \text{"Blue"}) \text{ And } ((\text{Count} * 2) < 21))$

แบบฟอร์ม (Form)

ตามที่ได้กล่าวไว้ในบทที่ 1 แล้วว่าแบบฟอร์มเปรียบเสมือนกับเป็นผ้าใบที่จะใช้สร้างแอปพลิเคชันลงไป แบบฟอร์มแต่ละแบบจะตรงกันกับวินโดว์ที่อยู่ในแอปพลิเคชันขณะดำเนินงาน

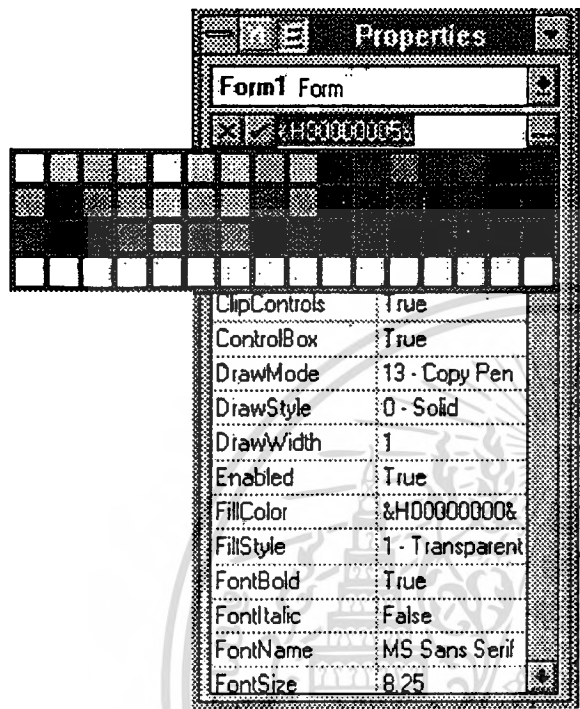
คุณสมบัติ (Property)

คุณสมบัติหลาย ๆ อย่างสามารถมีผลต่อลักษณะปรากฏและพฤติกรรมของแบบฟอร์มในเวลา que แสดงแบบฟอร์มนั้นออกมา แต่ในที่นี้แสดงเฉพาะคุณสมบัติของแบบฟอร์มที่นิยมใช้บ่อยที่สุดการตั้งคุณสมบัติอาจทำได้โดยใช้วินโดว์ Properties ในขณะที่กำลังออกแบบแอปพลิเคชันหรืออาจทำได้โดยการเขียนรหัสโปรแกรมที่ตั้งคุณสมบัติในขณะที่โปรแกรมดำเนินงาน

AutoRedraw คุณสมบัติ AutoRedraw ทำหน้าที่ควบคุมดูแลว่าจะสร้างภาพลักษณะจอภาพ (screen image) ขึ้นอีกครั้งอย่างไร เมื่อสวิตช์กลับไปยังแบบฟอร์มของ Visual Basic หลังจากทำงานในวินโดว์อื่น ๆ ที่ปกคลุมแบบฟอร์มไว้ Visual Basic ก็จะวาดกราฟิกใด ๆ บนแบบฟอร์มนั้นขึ้นใหม่โดยอัตโนมัติถ้าคุณตั้งคุณสมบัติ AutoRedraw เป็น True แต่ถ้าตั้งคุณสมบัตินี้เป็น False Visual Basic ก็จะต้องเรียกกระบวนการอื่นเพื่อทำงานขึ้นนี้แทนที่

BackColor คุณสมบัตินี้ใช้กำหนดสีของภาวะส่วนหลัง (background) เมื่อเลือก BackColor จากรายการคุณสมบัติที่อยู่ในวินโดว์ Properties จะพบค่ากำหนดโดยปริยายเป็น &H80000005& ซึ่งเป็นค่าคงตัวฐานสิบหกที่นิยามสีของ Visual Basic (ค่าคงตัวเหล่านี้จะถูกบรรจุอยู่ในไฟล์ CONSTANT.TXT) จะให้รายละเอียดเกี่ยวกับค่ากำหนดสีนี้ อย่างโดยละเอียดมากขึ้น แต่ในการตั้งคุณสมบัติ BackColor ไม่จำเป็นต้องรู้รายละเอียด

เซตดิงบ็อกซ์ในวินโดว์ Properties และ Visual Basic จะแสดงชุดอุปกรณ์สีตามที่แสดงไว้ในที่นี้



เลือกสีจากชุดอุปกรณ์โดยคลิกปุ่มเมาส์บนช่องสีเหลี่ยมที่เหมาะสม หรือคลิกปุ่มเมาส์บนปุ่มที่อยู่ด้านข้างของเซตดิงบ็อกซ์เพื่อปิดชุดอุปกรณ์นั้น

BorderStyle เราสามารถตั้งคุณสมบัตินี้ให้เป็นค่ากำหนดที่นิยามไว้แล้วอย่างใดอย่างหนึ่งในสื่อทางการตั้งคุณสมบัติ BorderStyle ทำได้เฉพาะในเวลาออกแบบเท่านั้น ค่ากำหนดไม่มีผลต่อการแสดงผลของแบบฟอร์มในเวลาออกแบบ แต่จะเปลี่ยนแปลงการแสดงผลของวินโดว์ในเวลาดำเนินงานดังนี้

- | | |
|-----------------|--|
| 0- None | วินโดว์ไม่มีขอบเขต |
| 1- Fixed Single | วินโดว์มีขนาดตายตัว (ระบุในเวลาออกแบบ) พร้อมกับมีขอบเขตเส้นเดียว |
| 2- Sizable | วินโดว์เปลี่ยนแปลงขนาดได้พร้อมกับมีขอบแบบเส้นคู่มาตรฐาน |

3- Fixed Double

วินโดว์มีขนาดตายตัว (ระบุในเวลาออกแบบ) และมีขอบแบบเส้นคู่

Caption	คุณสมบัตินี้กำหนดข้อความที่แสดงในไต่เลบารของแบบฟอร์มหรือข้อความที่แสดงในหรือถัดจากตัวควบคุมตัวใดตัวหนึ่ง
ControlBox	คุณสมบัติ Boolean คุณสมบัตินี้ควรจะต้องให้เป็น True ถ้าต้องการให้แสดงบ็อกซ์สำหรับเมนู Control ออกมาตรงมุมด้านบนซ้ายของวินโดว์ เราสามารถดัดแปลงคุณสมบัตินี้เฉพาะในเวลาออกแบบเท่านั้น การดัดแปลงคุณสมบัติ ControlBox ให้เป็น True จะไม่มีผลใด ๆ ถ้าตั้ง BorderStyle ให้เป็น 0- None
Enabled	ออปเจ็กต์ทุก ๆ ตัวของ Visual Basic มีคุณสมบัติ Enabled ที่สามารถตั้งให้เป็น True หรือ False ซึ่งเป็นการทำออปเจ็กต์ให้อยู่ในสภาพใช้งานได้ หรืออยู่ในสภาพใช้งานไม่ได้ ในกรณีของแบบฟอร์มหนึ่ง ๆ ตามปกติตั้งคุณสมบัตินี้ให้เป็น True แต่อาจตั้งคุณสมบัตินี้ให้เป็น False เพื่อป้องกันไม่ให้มีการส่งอีเวนต์ของเมาส์หรือคีย์บอร์ดไปยังแบบฟอร์ม
FantBold, FontItalic, FontStrikethru, FontUnderline	เราสามารถตั้งคุณสมบัติ Boolean เหล่านี้ให้เป็น True ถ้าต้องการให้ข้อความที่พิมพ์ออกไปที่แบบฟอร์มปรากฏอยู่ในชนิดตัวหนา ตัวเอน ตัวที่มีเส้นขีดขวางตลอด หรือตัวที่มีเส้นใต้ (หรือที่มีแอตทริบิวต์เหล่านี้รวมเข้าด้วยกัน) ค่ากำหนดใหม่นี้มีผลต่อเฉพาะอักขระที่พิมพ์ หลังจากที่ได้เปลี่ยนแปลงคุณสมบัติเหล่านี้อย่างใดอย่างหนึ่งแล้วเท่านั้น
FontName	คุณสมบัตินี้จะตั้งให้เป็นชื่อของฟอนต์ที่จะนำไปใช้สำหรับพิมพ์ไปที่แบบฟอร์ม ถ้าเปลี่ยนแปลงฟอนต์ ข้อความใด ๆ ก็ตามที่เขียนไปที่แบบฟอร์มแล้วจะไม่ถูกเปลี่ยนแปลงแต่จะส่งผลต่อการพิมพ์ที่เกิดขึ้นในช่วงต่อมา เช่น ถ้าเอ็กซีคิวต์รหัสต่อไปนี้ อักษร C จะพิมพ์ออกมาในฟอนต์ Courier และอักขระที่เหลือทั้งหมดจะพิมพ์อยู่ใน Time New Roman

FontName = "Times New Roman"

Print "AB"

FontName = "Courier New"

Print "C"

FontName = “Times New Roman”

Print “DE”

FontSize	คุณสมบัตินี้ทำให้สามารถกำหนดขนาด (ในหน่วยจุดภาพ) สำหรับข้อความที่จะพิมพ์ออกไปที่แบบฟอร์ม (พอยต์ เป็นหน่วยการเรียงพิมพ์มาตรฐานที่ใช้วัดขนาดของชนิด โดย 1 พอยต์เท่ากับประมาณ 1/72 นิ้ว) ค่ากำหนดไม่ได้เปลี่ยนแปลงขนาดของข้อความใด ๆ ที่ได้พิมพ์ออกมาแล้ว
ForeColor	คุณสมบัตินี้นิยามสีของข้อความหรือกราฟิกส่วนหน้า (foreground) การตั้งคุณสมบัตินี้ทำได้โดยการเลือกสีจากชุดอุปกรณ์สีด้วยแนวทางเดียวกันกับที่ใช้ตั้งคุณสมบัติ BackColor ข้อความสั่ง Print ทั้งหมดจะเขียนข้อความโดยใช้สีส่วนหน้า
Height, Width	คุณสมบัติเหล่านี้กำหนดความสูงและความกว้างของแบบฟอร์มที่วัดอยู่ในหน่วยที่เรียกว่า ทวิพ หรือเศษหนึ่งส่วนสี่สิบของพอยต์ (ทวิพเทียบเท่ากับ 1/1440 นิ้ว)
Icon	เราสามารถตั้งคุณสมบัตินี้ให้เป็นชื่อของไฟล์ที่บรรจุไอคอน ไอคอนนี้จะไปแทนแบบฟอร์มเมื่อลดขนาดแบบฟอร์ม ในการตั้งคุณสมบัตินี้ในเวลาออกแบบ ให้เลือก Icon จากรายการคุณสมบัติในวินโดว์ Properties และหลังจากนั้นกดปุ่มเมาส์บนปุ่มไขว้ปลาที่ปรากฏอยู่ด้านข้างของเซตคีย์บอร์ด ในไดอะล็อกบ็อกซ์ที่เกิดขึ้น เลือกไฟล์ไอคอน ถ้าต้องการคุณสมบัติในเวลาดำเนินงาน จะต้องใช้ฟังก์ชัน LoadPicture หรือ กำหนดค่าของคุณสมบัติ Icon ของแบบฟอร์มอื่น ๆ
MaxButton, MinButton	เราควรจะต้องคุณสมบัติ Boolean เหล่านี้เป็น True ถ้าต้องการให้แสดงปุ่มขยายขนาดและปุ่มลดขนาด ที่ตรงมุมด้านบนขวาของวินโดว์ คุณสมบัติทั้งสองนี้จะถูกมองข้ามถ้าได้ตั้งคุณสมบัติ BorderStyle ให้เป็น 0- None ควรทราบไว้ด้วยว่าปุ่มขยายขนาดและปุ่มลดขนาดไม่ได้หายไปจากแบบฟอร์มในเวลาออกแบบ แม้แต่เมื่อคุณตั้งคุณสมบัติ MaxButton และ MinButton ให้เป็น False ผลของค่ากำหนดเหล่านี้จะไม่ปรากฏในวินโดว์จนกว่าถึงเวลาดำเนินงาน
Name	คุณสมบัตินี้นิยามชื่อของฟอร์มในรหัสโปรแกรม ชื่อนี้ไม่สามารถเปลี่ยนแปลงในเวลาดำเนินงาน
Picture	เราสามารถตั้งคุณสมบัตินี้เพื่อแสดงว่าจะแสดงภาพแบบแผนที่ปิด (bit map) ในวินโดว์เสมอ ในการตั้งคุณสมบัตินี้ในเวลาออกแบบ ให้เลือกคุณสมบัตินี้จากรายการ

คุณสมบัติ และกดปุ่มเมาส์บนปุ่มไขว้ปลาที่ปรากฏอยู่ตรงด้านข้างของเซตคิงบ็อกซ์ในวินโดว์ Properties ในไดอะล็อกบ็อกซ์ ที่เกิดขึ้น ให้เลือกไฟล์ที่บรรจุภาพที่ต้องการแสดงในการเปลี่ยนแปลงคุณสมบัตินี้ในเวลาดำเนินงานต้องใช้ฟังก์ชัน

LoadPicture

Top, Left	คุณสมบัติเหล่านี้ควบคุมตำแหน่งของแบบฟอร์มบนจอภาพโดยการนิยามตำแหน่งส่วนบนสุดของวินโดว์ และพิกัดด้านซ้ายสุดที่วัดในหน่วยทวิพอยน์ทโดยปริยาย
Visible	ค่ากำหนดของคุณสมบัติ Boolean นี้แสดงถึงว่าจะให้ฟอร์มวินโดว์เห็นได้หรือไม่ เห็นการตั้งคุณสมบัตินี้ไปเป็น False จะซ่อนไม่ให้เห็นแบบฟอร์ม
WindowState	คุณสมบัตินี้กำหนดค่าให้แสดงวินโดว์ออกมาในสถานะปกติ (ค่ากำหนดเป็น 0) ในสถานะลดขนาดเป็นไอคอน (ค่ากำหนดเป็น 1) หรือในสถานะขยายขนาด (ค่ากำหนดเป็น 2)
อีเวนต์ (Event)	Click, DbClick และ Load เป็นอีเวนต์ที่พบบ่อยที่สุดสำหรับแบบฟอร์มที่จะประมวล (อีเวนต์ KeyPress และกระบวนการลากแล้วปล่อย (drag and drop))
Click	เมื่อผู้ใช้กดปุ่มเมาส์บนแบบฟอร์ม Visual Basic เรียกกระบวนการ Form_Click ถ้าผู้ใช้กดปุ่มเมาส์บนออบเจกต์ที่บรรจุอยู่ในแบบฟอร์ม อีเวนต์ Click ก็จะไม่ถูกส่งผ่านไปให้แก่แบบฟอร์มแม้แต่เมื่อออบเจกต์อื่น ๆ ไม่ได้ประมวลอีเวนต์นั้น
DbClick	อีเวนต์นี้จะได้รับถ้าผู้ใช้กดปุ่มเมาส์สองครั้งต่อเนื่องกันอย่างรวดเร็วที่จุดบางจุดบนแบบฟอร์ม โดยแท้จริงแล้วกระบวนการนี้สับเปลี่ยนอีเวนต์สองอีเวนต์ โดยอีเวนต์ Click สำหรับการกดปุ่มเมาส์ครั้งแรก และอีเวนต์ DbClick สำหรับการกดปุ่มเมาส์ครั้งที่สอง
Load	เหตุการณ์นี้เกิดขึ้นอย่างโดยอัตโนมัติ เมื่อใดก็ตามที่บรรจุแบบฟอร์ม เช่น เมื่อเริ่มใช้งานแอปพลิเคชัน กระบวนการอีเวนต์ Load เป็นกระบวนการที่ใช้สะดวกสำหรับการกำหนดค่าเริ่มต้นของคุณสมบัติและตัวแปรในเวลาเริ่มใช้งานโปรแกรม

กระบวนการและกลวิธี (Method)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กลวิธีของแบบฟอร์มที่น่าสนใจมากที่สุดคือ กลวิธีกราฟิก เนื่องจากกลวิธีนี้ค่อนข้างซับซ้อน ดังนั้นจึงได้ใช้บททั้งบทกล่าวเกี่ยวกับกราฟิกในหนังสือเล่มนี้ สำหรับในตอนนี้จะพิจารณาการกลวิธีเพียงไม่กี่อย่าง (และหนึ่งฟังก์ชัน) เพื่อให้สามารถเริ่มใช้งานได้

Cls กลวิธีนี้ลบล้างแบบฟอร์มของกราฟิกและข้อความทั้งหมด วากยสัมพันธ์อย่างสมบูรณ์สำหรับกลวิธี Cls เป็นดังต่อไปนี้

`[formName].Cls`

LoadPicture เราสามารถใช้ฟังก์ชันนี้เพื่อตั้งคุณสมบัติ Picture ของแบบฟอร์ม วากยสัมพันธ์ของฟังก์ชันเป็นดังนี้

`LoadPicture ([fileName])`

ชื่อไฟล์จะต้องเป็นนิพจน์สตริงที่ระบุไฟล์ที่ประกอบไปด้วยไอคอน บิตแมปของ Windows หรือไฟล์เมตาของ Windows (ชื่อไฟล์เหล่านี้ตามปกติมีส่วนขยายเป็น ICO, BMP และ WMF) ถ้าไม่ได้กำหนดชื่อไฟล์ไว้ LoadPicture จะส่งคืนภาพว่าง ๆ

Print กลวิธีนี้เขียนข้อความไปยังแบบฟอร์ม ใช้วากยสัมพันธ์ดังนี้

`[formName].Print[[expression]][,;]]...`

แต่ละนิพจน์จะตามหลังด้วยอ้อมภาค (;) หรือจุลภาค (,) ถ้าใช้อ้อมภาคจะทำให้ผลส่งออกของนิพจน์อยู่หลังอักขระตัวสุดท้ายที่เขียนในทันที แต่ถ้าใช้จุลภาค ตำแหน่งผลส่งออกบนแบบฟอร์มจะถูกตั้งระยะให้อยู่ที่คอลัมน์ถัดไป (คอลัมน์จะนิยามทุก ๆ อักขระ 14 ตัว) ถ้านิพจน์สุดท้ายในข้อความสั่ง Print ลงท้ายด้วยอ้อมภาคหรือจุลภาค จะไม่มีการเขียนอักขระปิดแคโรไปที่แบบฟอร์ม มิฉะนั้นแล้วจะตั้งตำแหน่งผลส่งออกให้อยู่ที่บรรทัดถัดไป ต่อไปนี้เป็นตัวอย่างของการใช้กลวิธี Print ในวินโดว์ Debug

บ็อกซ์ภาพและภาพลักษณะ (Image and Picture)

ตัวควบคุมบ็อกซ์ภาพและภาพลักษณะทำให้คุณสามารถวางสารสนเทศทางกราฟิกไว้ในตำแหน่งเฉพาะบนแบบฟอร์ม เครื่องมือสร้างบ็อกซ์ภาพและเครื่องมือสร้างภาพลักษณะจากทูลบาร์ได้แสดงเอาไว้ในรูปที่ 2.7 และ 2.8 บ็อกซ์ภาพจะมีความยืดหยุ่นมากกว่า และด้วยเหตุนี้จึงต้องการหน่วยความจำและการประมวลผลในปริมาณที่มากกว่า บ็อกซ์ภาพเหมาะสมอย่างยิ่งสำหรับสภาพแวดล้อมไคนาติก เช่น เมื่อคุณกำลังวาดกราฟิกเข้าไปที่จอภาพโดยตรงในขณะที่โปรแกรมกำลังดำเนินงานหรือการทำไอคอนให้เคลื่อนไหวได้ ด้วยการโยกย้ายไอคอนไปรอบ ๆ บนจอภาพ ออบเจ็กต์ภาพลักษณะที่ดีที่

สุดสำหรับสถานการณ์สดิต นั่นคือเมื่อไม่มีการคัดแปลงบิตแมปหรือไอคอนที่คุณสร้างและวางเอาไว้บนจอภาพ



รูปที่ 2.7

เครื่องมือสร้างบ็อกซ์ภาพ



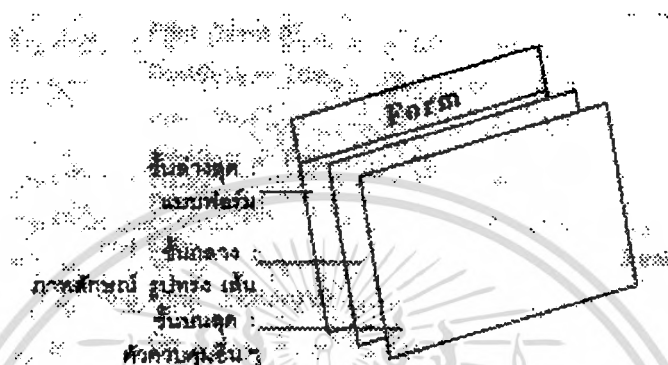
รูปที่ 2.8

เครื่องมือสร้างภาพลักษณ์

คุณสมบัติ

ตัวควบคุมบ็อกซ์ภาพและภาพลักษณ์มีคุณสมบัติ Enabled, Height, Left, Name, Picture, Top, Visible และ Width คุณสมบัติเหล่านี้ส่วนใหญ่แล้วจะประพฤติปฏิบัติเหมือนกันกับที่ได้บรรยายไว้ในตอนก่อนหน้านี้นั้นแบบฟอร์ม ยกเว้นแต่ว่าจะวัดพิกัดสำหรับคุณสมบัติ Height, Left, Top และ Width โดยเทียบกับตำแหน่งของออบเจกต์ที่อยู่ในแบบฟอร์ม ไม่ได้วัดในรูปของพิกัดจอภาพสัมบูรณ์ (ตัวควบคุมทั้งหมดที่สร้างขึ้นจากทูลบาร์จะมีลักษณะเช่นนี้) คุณสามารถตั้งคุณสมบัติ Picture ให้แสดงบิตแมปหรือไอคอน โดยในเวลาออกแบบให้ใช้วินโดว์ Properties และคลิกเลือกบ็อกซ์ Load Picture และในเวลาดำเนินงานคุณจำเป็นต้องใช้ฟังก์ชัน Load Picture บ็อกซ์ภาพ ซึ่งประพฤติปฏิบัติเกือบเหมือนกันกับแบบฟอร์มเล็ก ๆ ที่อยู่ในแบบฟอร์มหลักมีคุณสมบัติต่อไปนี้ที่ไม่พบอยู่ในตัวควบคุมภาพลักษณ์ นั่นคือ AutoRedraw, FontBold, FontItalic, FontName, FontSize, FontUnderline คุณสมบัติเหล่านี้ประพฤติปฏิบัติในทำนองเดียวกันที่มีไว้สำหรับแบบฟอร์ม ตัวควบคุมภาพลักษณ์ครอบครองคุณสมบัติอย่างหนึ่งที่ไม่อยู่ในบ็อกซ์ภาพ นั่นคือคุณสมบัติ Stretch โดยปริยายแล้วจะตั้ง Stretch ให้เป็น False ซึ่งทำให้ตัวควบคุมภาพลักษณ์เปลี่ยนขนาดตัวมันเองเพื่อให้พอเหมาะกับภาพที่ตัวควบคุมนั้นบรรจุอยู่ ถ้าคุณตั้ง Stretch ให้เป็น True ภาพจะเปลี่ยนขนาดให้พอเหมาะกับตัวควบคุม ตัวควบคุมบ็อกซ์ภาพและภาพลักษณ์ยังมีลักษณะอีกประการหนึ่งที่แตกต่างกัน จอแสดงผลของแบบฟอร์มนั้นจะประกอบกันขึ้นมาจากชั้นสามชั้นเช่นที่แสดงไว้ในรูปที่ 2.9 สารสนเทศทั้งหมดที่แสดงบน

แบบฟอร์มโดยตรง จะปรากฏอยู่บนชั้นล่างสุด สารสนเทศจากตัวควบคุมกราฟิกสามตัว (ตัวควบคุมภาพลักษณ์รูปทรง และเส้น) จะปรากฏอยู่ตรงชั้นกลาง และออบเจ็กต์อื่น ๆ ทั้งหมดจะอยู่บนชั้นบนสุดด้วยเหตุนี้ภาพลักษณ์จึงอาจบดบังข้อความที่พิมพ์ออกมาบนแบบฟอร์ม ซึ่งโดยต่อเนื่องกันบ็อกซ์ภาพก็อาจบดบังภาพลักษณ์ได้เช่นกัน



รูปที่ 2.9

ชั้นของจอแสดงผลของแบบฟอร์ม

อีเวนต์

ทั้งตัวควบคุมบ็อกซ์ภาพและภาพลักษณ์สามารถรับอีเวนต์ Click และ DblClick ซึ่งเป็นอีเวนต์ที่ง่ายที่สุดและพบบ่อยที่สุด

กระบวนการและกลวิธี

คุณสามารถเรียกฟังก์ชัน `LoadPicture` เพื่อใช้ตั้งคุณสมบัติ `Picture` ของตัวควบคุมบ็อกซ์ภาพหรือภาพลักษณ์ บ็อกซ์ภาพยังสนับสนุนกลวิธี `Cls` และ `Print` รวมทั้งกลวิธีกราฟิก

เลเบล (Label)

เลเบลประกอบไปด้วยพื้นที่ซึ่งคุณสามารถนำเสนอสารสนเทศประเภทข้อความที่ผู้ใช้ไม่สามารถจะแก้ไขได้ การตั้งเนื้อหาของเลเบลทำได้โดยการดัดแปลงคุณสมบัติ `Caption` ของเลเบล คุณไม่สามารถพิมพ์หรือวาดเข้าไปในเลเบล คุณสร้างเลเบลโดยอาศัยเครื่องมือสร้างเลเบลจากทูลบาร์ของ Visual Basic เช่นที่แสดงในรูปที่ 2.10



รูปที่ 2.10

เครื่องมือสร้างเลเบล

คุณสมบัติ

เลเบลมีคุณสมบัติ FontBold, FontItalic, FontName, FontSize, FontUnderline, Height, Left, Name, Top, Visible และ Width ซึ่งดำเนินการด้วยแนวทางที่เป็นมาตรฐาน คุณสมบัติเพิ่มเติมบางอย่างได้บรรยายไว้ในที่นี้แล้ว

Alignment	คุณสมบัตินี้กำหนดว่าจะวางข้อความจำหน่ายของเลเบลอย่างไร โดยปรีชาแล้วจะตั้งคุณสมบัติ Alignment ให้เป็น 0 ซึ่งกำหนดให้ข้อความจำหน่ายอยู่ชิดซ้าย คุณยังสามารถตั้ง Alignment ให้เป็น 1-Right Justify หรือเป็น 2-Center
AutoSize	ถ้าตั้งคุณสมบัตินี้ให้เป็น True ก็จะเปลี่ยนขนาดเลเบลโดยอัตโนมัติเพื่อให้พอเหมาะ กับข้อความที่กำหนดไว้ด้วยคุณสมบัติ Caption ถ้าตั้ง AutoSize ให้เป็น False เลเบล จะยังคงมีขนาดตามที่นิยามในเวลาออกแบบ ในกรณีนั้นถ้าข้อความจำหน่ายยาวเกินไป ข้อความนั้นจะถูกคัดออกเพื่อให้พอเหมาะ กับเลเบล
BorderStyle	คุณสมบัตินี้สามารถตั้งให้เป็น 0 (ค่าปรีชา) สำหรับกรณีไม่มีขอบ หรือเป็น 1 สำหรับขอบแบบเส้นเดี่ยว
Caption	คุณสมบัตินี้ประกอบไปด้วยข้อความที่จะแสดงออกมาในฟิลด์ของเลเบลหรือพื้นที่ของจอภาพ
Enabled	ตามปกติจะตั้งคุณสมบัตินี้ให้เป็น True แต่คุณอาจตั้งให้เป็น False เพื่อทำข้อความให้เป็นสีเทา และทำให้ไอเวนต์เมาส์ต่าง ๆ ใช้การไม่ได้

ไอเวนต์

เลเบลตอบสนองกับไอเวนต์ Click และ DblClick ในทำนองเดียวกันกับที่ตัวควบคุมบ็อกซ์ภาพภาพ ลักษณะ และแบบฟอร์มสามารถทำได้

กระบวนการและกลวิธี

เลเบลออกแบบก็ทำได้ไว้เพื่อใช้แสดงบล็อกของข้อความขนาดเล็ก ด้วยเหตุที่สามารถนิยามข้อความโดยเพียงแต่ตั้งคุณสมบัติ Caption ดังนั้นจึงไม่มีกลวิธีที่เกี่ยวข้องกับเลเบลที่จะมีประโยชน์หรือจำเป็นต่อผู้เริ่มต้น

เท็กซ์บ็อกซ์ (Text Box)

คุณได้ทดลองใช้งานเท็กซ์บ็อกซ์มาแล้ว โคนการใช้เครื่องมือสร้างเท็กซ์บ็อกซ์เช่นที่แสดงไว้ในรูปที่ 2.11 คุณเพียงวางเท็กซ์บ็อกซ์ไว้บนแบบฟอร์ม ก็จะทำให้สามารถสร้างบริเวณของจอภาพที่ผู้ใช้จะสามารถป้อนข้อความเข้าไปได้



รูปที่ 2.11

เครื่องมือสร้างเท็กซ์บ็อกซ์

คุณสมบัติ

เท็กซ์บ็อกซ์มีคุณสมบัติมาตรฐานคือ BorderStyle, Enabled, FontBold, FontItalic, FontName, FontSize, FontUnderline, Height, Left, Name, Top, Visible และ Width บวกกับคุณสมบัติเพิ่มเติมต่อไปนี้

- MaxLength** คุณสมบัตินี้มักจะตั้งให้เป็น 0 (โดยปริยาย) ซึ่งแสดงว่าเท็กซ์บ็อกซ์จะรับอักขระได้มากเท่าที่ผู้ใช้พิมพ์เข้าไป อย่างไรก็ตาม ถ้าคุณตั้ง MaxLength ให้เป็นค่าที่ไม่ใช่ศูนย์ Visual Basic ก็จะจำกัดสิ่งที่ผู้ใช้ป้อนเข้าไปให้มีจำนวนของอักขระตามที่ระบุไว้
- MultiLine** บ่อยครั้งที่คุณจะต้องคุณสมบัติให้เป็น False ซึ่งจำกัดสิ่งที่ผู้ใช้ป้อนเข้าไปให้เป็นข้อความบรรทัดเดียว เมื่อตั้งคุณสมบัติ MultiLine ให้เป็น True ผู้ใช้สามารถกดแป้น Enter เพื่อวางการปิดแครวไว้ในข้อความ และป้อนข้อความต่อไปบนบรรทัดถัดไป
- PasswordChar** คุณสมบัตินี้ควบคุมว่าเท็กซ์บ็อกซ์เป็นฟิลด์รหัสผ่านหรือไม่ เมื่อผู้ใช้พิมพ์ข้อความเข้าไปในฟิลด์รหัสผ่าน อักขระที่ปรากฏขึ้นบนจอภาพจะแตกต่างไปจากอักขระที่ได้พิมพ์เข้าไป โดยปริยายแล้วจะตั้งคุณสมบัตินี้เป็นสตริงว่าง ซึ่งหมายถึงว่าผู้ใช้จะเห็นอักขระที่พวกเขากำลังป้อนเข้าไป ถ้าตั้งคุณสมบัติ PasswordChar ให้เป็นอักขระ

เช่นคอกจัน ก็จะแสดงอักขระนั้นออกมาสำหรับอักขระทุก ๆ ตัวที่ผู้พิมพ์เข้าไปถึงตอนนี้แล้วอย่าเพิ่งสับสน เนื้อหาที่แท้จริงของเท็กซ์บ็อกสนั้นจะถูกตั้งให้เป็นข้อความที่ผู้พิมพ์เข้าไป คุณสมบัตินี้มีผลต่อการแสดงผลเท่านั้น

ScrollBars	คุณสามารถตั้งคุณสมบัตินี้ให้เป็น 0 (ไม่มีสกรอลล์บาร์ในเท็กซ์บ็อกซ์) 1. (มีเฉพาะสกรอลล์บาร์ตามแนวนอน) 2. (มีเฉพาะสกรอลล์บาร์ตามแนวตั้ง) หรือ 3. (มีสกรอลล์บาร์ทั้งแนวนอนและแนวตั้ง)
SelLength	คุณสมบัตินี้ประกอบไปด้วยจำนวนของอักขระที่เลือกเอาไว้ล่าสุด ค่าของคุณสมบัติจะเปลี่ยนแปลงเมื่อผู้ใช้เลือกข้อความในเท็กซ์บ็อกซ์ คุณยังสามารถจะเปลี่ยนการเลือกผ่านทางโปรแกรมด้วยการตั้งคุณสมบัตินี้ให้เป็นค่าจำนวนเต็มในรหัสของคุณ ถ้า SelLength เป็น 0 นั้นหมายถึงไม่ได้เลือกอักขระไว้ คุณสามารถ เข้าถึงคุณสมบัตินี้และคุณสมบัติ SelStart และ SelText ได้เฉพาะในเวลาดำเนินงานเท่านั้น
SelStart	การตั้งคุณสมบัติบอกให้คุณทราบว่าการเลือกในขณะนั้นเริ่มต้นที่ใด การตั้งให้เป็น 0 แสดงว่าการเลือก (หรือจุดแทรก) เริ่มต้นก่อนอักขระตัวแรก การตั้งเป็น 1 แสดงว่าการเลือกเริ่มต้นก่อนอักขระตัวที่สอง และเป็นเช่นนี้ต่อเนื่องไป คุณสามารถคัดแปลงคุณสมบัตินี้ผ่านทางโปรแกรมได้เช่นกัน
SelText	คุณสมบัตินี้ประกอบไปด้วยสตริงข้อความที่ได้เลือกไว้ในขณะนั้น ถ้าไม่ได้เลือกข้อความใด ๆ ไว้เลย คุณสมบัติก็จะประกอบไปด้วยสตริงว่าง ถ้าคุณตั้ง SelText ในรหัสโปรแกรมของคุณ จะเป็นการแทนที่ข้อความที่เลือกไว้ในเท็กซ์บ็อกซ์ด้วยการตั้งค่าของ SelText เช่น ถ้าเท็กซ์บ็อกซ์หนึ่งชื่อว่า Text1 ประกอบไปด้วย บรรทัด "Frankly, my deere, I don't give a damn" และเลือกคำว่า deere ไว้ คุณสามารถแก้ไขประโยคได้โดยการเอ็กซิคิวต์ข้อความสั่ง <code>Text1.SelText = "dear"</code> การทำเช่นนี้ยังเปลี่ยนแปลงค่าของ SelLength ด้วยเช่นกัน อย่างไรก็ตามด้วยเหตุที่การเลือกเริ่มต้นที่ตำแหน่งเดียวกัน ดังนั้น SelStart จึงไม่มีผลใด ๆ ทั้งสิ้น
Text	โปรแกรมอ่านคุณสมบัตินี้เพื่อดูว่าผู้ใช้อะไรเข้าไป โปรแกรมยังสามารถคัดแปลงคุณสมบัติ Text เพื่อเปลี่ยนข้อความที่แสดงออกมา

อีเวนต์

เท็กซ์บ็อกซ์ไม่สนับสนุนอีเวนต์เมาส์ แต่จะสามารถตรวจจับการเคาะแป้นที่ผู้ใช้ป้อนเข้าไป เราจะศึกษาอีเวนต์ของคีย์บอร์ดในภายหลัง

- Change** การส่งสัญญาณมาที่อีเวนต์นี้จะเกิดขึ้นเมื่อมีการเปลี่ยนแปลงคุณสมบัติ Text ของเท็กซ์บ็อกซ์ ไม่ว่าจะเป็นจากผู้ใช้งานซึ่งป้อนสารสนเทศใหม่เข้าไป หรือจากโปรแกรมซึ่งสามารถตั้งค่าใหม่ให้แก่คุณสมบัติ Text เช่น ถ้าผู้ใช้พิมพ์คำว่า *Hello* จะส่งสัญญาณไปยังอีเวนต์ Change ห้าครั้ง โดยแต่ละครั้งสำหรับอักขระแต่ละตัวที่พิมพ์ เข้าไป
- LostFocus** การส่งสัญญาณมาที่อีเวนต์นี้จะเกิดขึ้นเมื่อผู้ใช้งานกระดกเมาส์ออกจากเท็กซ์บ็อกซ์ หรือใช้เมาส์เลือกออกเจ็ทต์อื่น ๆ ในแบบฟอร์ม ถ้าคุณต้องการตรวจสอบเนื้อหาของคุณสมบัติ Text การทำเป็นส่วนหนึ่งของกระบวนการงานอีเวนต์ LostFocus จะมีประสิทธิภาพมากกว่าการทำเป็นส่วนหนึ่งของกระบวนการงานอีเวนต์ Change

กระบวนการและกลวิธี

กลวิธีที่เกี่ยวข้องกับเท็กซ์บ็อกซ์ที่มีประโยชน์ที่สุดคือ *SetFocus* ซึ่งจะวางเคอร์เซอร์ไว้ในเท็กซ์บ็อกซ์เฉพาะ เมื่อคุณสร้างแบบฟอร์มที่มีเท็กซ์บ็อกซ์หลาย ๆ บ็อกซ์ คุณสามารถใช้กลวิธีนี้เพื่อชี้นำความสนใจของผู้ใช้ไปที่บ็อกซ์ที่สำคัญที่สุด วากยสัมพันธ์จะเป็นดังนี้

[object.]SetFocus

ปุ่มคำสั่ง เช็บบ็อกซ์ และปุ่มออฟชั่น (Command Button , Check Box and Option)

ออบเจ็กต์สามตัวนี้ของ Visual Basic มีลักษณะที่คล้ายกันเมื่อพิจารณาในทฤษฎีของนักเขียนโปรแกรม อย่างไรก็ตาม ผู้ใช้น้องเห็นสามปุ่มนี้มีความแตกต่างกัน และนำไปใช้ในจุดประสงค์ที่แตกต่างกัน ตามปกติเมื่อคลิกปุ่มเมาส์บนปุ่มคำสั่งมักจะทำให้มีการทำงานอย่างใดอย่างหนึ่งเกิดขึ้น (นั่นคือเอ็กซีคิวต์คำสั่งหนึ่ง ๆ) ในขณะที่เช็บบ็อกซ์และปุ่มออฟชั่นแสดงถึงสถานะและให้ผู้ใช้สามารถเปลี่ยนแปลงสถานะเหล่านั้น ผู้ใช้กดปุ่มเมาส์บนเช็บบ็อกซ์ เพื่อเลือกหรือยกเลิกการเลือกคุณลักษณะบางประการ โดยเมื่อ X ปรากฏอยู่ในเช็บบ็อกซ์ แสดงว่าเลือกคุณลักษณะนั้นหรือเปิดใช้งานคุณลักษณะนั้น เช็บบ็อกซ์แต่ละบ็อกซ์ดำเนินการแยกต่างหากกัน ส่วนปุ่มออฟชั่นจะปรากฏอยู่เป็นชุด และให้ผู้ใช้เลือกออฟชั่นตัวใดตัวหนึ่งจากชุดของออฟชั่นนั้นเมื่อเปิดใช้งานปุ่มออฟชั่นใดปุ่มหนึ่ง ปุ่มอื่น ๆ จะปิดการใช้ (ในบางครั้งจะเรียกปุ่มออฟชั่นนี้ว่า *ปุ่มวิทยุ (radio button)*)

เครื่องมือจากทูลบาร์ที่นำมาใช้สร้างออบเจกต์ทั้งสามชนิดนี้ของ Visual Basic ได้นำมาแสดงไว้ในรูปที่ 2.12, 2.13 และ 2.14



รูปที่ 2.12

เครื่องมือสร้างปุ่มคำสั่ง



รูปที่ 2.13

เครื่องมือสร้างเช็กรadioButton



รูปที่ 2.14

เครื่องมือสร้างปุ่มออปชัน

คุณสมบัติ

ปุ่มคำสั่ง เช็กรadioButton และปุ่มออปชันมีคุณสมบัติดังต่อไปนี้ Caption, Enabled, FontBold, FontItalic, FontName, FontSize, FontSize, FontUnderline, Height, Left, Name, Top, Visible และ Width นอกจากนี้คุณอาจพบคุณสมบัติที่เป็นประโยชน์ดังนี้

- Cancel** คุณสมบัตินี้จะใช้ได้เฉพาะกับปุ่มคำสั่งเท่านั้น เมื่อได้ตั้งคุณสมบัติ Cancel ของปุ่มคำสั่งให้เป็น True การกดแป้น Esc จะมีผลเช่นเดียวกันกับการกดปุ่มเมาส์บนปุ่มคำสั่งนั้น (เช่นคล้ายกับปุ่ม Cancel ในไดอะล็อกบ็อกซ์) ปุ่มคำสั่งเพียงปุ่มใดปุ่มหนึ่งเท่านั้นในแบบฟอร์มหนึ่ง ๆ ที่จะสามารถมีคุณสมบัติ Cancel ตั้งให้เป็น True ได้
- Default** คุณสมบัติ Default จะใช้ได้เฉพาะกับปุ่มคำสั่งเท่านั้น เมื่อตั้งคุณสมบัติให้เป็น True จะทำให้การกดแป้น Enter มีผลอย่างเดียวกันกับการกดปุ่มเมาส์บนปุ่มนั้น (คล้ายกับ

ปุ่ม OK ในไดอะล็อกบ็อกซ์ ปุ่มคำสั่งเพียงปุ่มใดปุ่มหนึ่งเท่านั้นในแบบฟอร์มหนึ่ง ๆ ที่จะสามารถมีคุณสมบัติ Default ตั้งให้เป็น True

Value คุณสมบัตินี้แสดงสถานะของเช็บบ็อกซ์หรือปุ่มออปชัน คุณสมบัติ Value ของปุ่มออปชันอาจตั้งให้เป็น True หรือ False อย่างใดอย่างหนึ่ง เมื่อตั้งคุณสมบัตินี้ให้เป็น True ปุ่มจะอยู่ในสภาพ “เปิด” นั่นคือจะทำบริเวณตรงกลางของปุ่มให้เด่นชัดขึ้น คุณสมบัติ Value ของเช็บบ็อกซ์อาจตั้งให้เป็น 0,1 หรือ 2 การตั้งให้เป็น 0 แสดงว่าบ็อกซ์นั้นไม่ได้ถูกตรวจสอบ การตั้งให้เป็น 1 แสดงว่าบ็อกซ์นั้นถูกตรวจสอบ และการตั้งให้เป็น 2 แสดงว่าทำบ็อกซ์นั้นให้เป็นสีเทา

อีเวนต์

ออบเจกต์ทั้งสามตัวนี้สามารถจะรับอีเวนต์ Click แม้ว่าคุณชอบที่จะใช้กระบวนการอีเวนต์ Click เฉพาะกับปุ่มคำสั่งเท่านั้น ปุ่มออปชันและเช็บบ็อกซ์จะแสดงสถานะของตัวเองออกมาโดยอัตโนมัติเมื่อคุณกดปุ่มเมาส์บนปุ่มเหล่านั้น โดยทั่วไปแล้วไม่จำเป็นต้องอาศัยการเขียนโปรแกรม

กระบวนการและกลวิธี

คุณไม่สามารถพิมพ์หรือวาดลงไปบนปุ่มคำสั่ง เช็บบ็อกซ์ หรือปุ่มออปชัน ไม่มีกลวิธีใด ๆ ที่พิจารณาผ่านมาแล้วจะสามารถนำมาใช้กับออบเจกต์ทั้งสามชนิดนี้

กรอบ (Frame)

กรอบมีไว้เพื่อใช้แบ่งแยกบ็อกซ์ของออบเจกต์อีกกลุ่มหนึ่งบนจอภาพ ในกรณีของปุ่มออปชัน กรอบยังมีผลต่อพฤติกรรมของปุ่มอีกด้วย ซึ่งจะแสดงให้เห็นในโครงการตัวอย่างที่ขกมา สำหรับออบเจกต์อื่น ๆ กรอบจะทำให้สามารถแยกกลุ่มออบเจกต์ออกจากกัน รวมทั้งให้คุณลักษณะใช้การได้/ใช้การไม่ได้ ในการวางกรอบบนแบบฟอร์ม ให้ใช้เครื่องมือสร้างกรอบจากทูลบาร์ของ Visual Basic เช่นที่แสดงอยู่ในรูปที่ 2.15



รูปที่ 2.15

เครื่องมือสร้างกรอบ

คุณสมบัติ

คุณสมบัติ Name ของกรอบใช้ในการระบุกรอบเข้าไว้ในรหัสของโปรแกรมของคุณ และคุณสมบัติ Caption ใช้นิยามข้อความซึ่งสามารถเห็นเป็นส่วนหนึ่งของกรอบ นอกจากนี้กรอบยังประกอบไปด้วยคุณสมบัติ Enabled, FontBold, FontItalic, FontName, FontSize, FontUnderling, Height, Left, Top, Visible และ Width

คุณสมบัติ Enabled ที่คุ้นเคยนั้นจะประพาศก่อนข้างแตกต่างไปสำหรับกรอบ ตามปกติมักจะตั้งคุณสมบัตินี้ให้เป็น True เมื่อใดที่ตั้งให้เป็น False จะทำให้ข้อความจำหน้าของกรอบกลายเป็นสีเทาและทำออบเจกต์ทั้งหมดที่อยู่ภายในกรอบอยู่ในสภาพใช้การไม่ได้ รวมทั้งเท็กซ์บ็อกซ์ตัวควบคุมปุ่มและออบเจกต์อื่น ๆ

อีเวนต์

กรอบไม่ตอบสนองต่ออีเวนต์เมาส์ หรือสิ่งที่ผู้ใช้ป้อนเข้าไป

กระบวนงานและกลวิธี

คุณไม่สามารถพิมพ์หรือวาดเข้าไปภายในกรอบ รวมทั้งไม่สามารถนำกรอบไปเกี่ยวข้องกับกราฟิก

ลิสต์บ็อกซ์ (List Box)

ลิสต์บ็อกซ์เป็นสิ่งที่ทำให้ผู้ใช้มีโอกาสเลือกออพชัน ลิสต์บ็อกซ์จะแสดงออพชันที่มีอยู่ออกมา และผู้ใช้สามารถเลือกชั้นรายการหนึ่ง ๆ หรือหน่วยรายการหนึ่ง ๆ ได้โดยการกดปุ่มเมาส์บนหน่วยนั้น ถ้ารายการทั้งหมดยาวเกินกว่าที่จะสามารถแสดงออกมาได้ในครั้งเดียว Visual Basic ก็จะเดิมสกรอลล์บาร์เข้าไปในลิสต์บ็อกซ์ การที่จะทำให้สามารถทำงานได้อย่างถูกต้องลิสต์บ็อกซ์ควรมีความสูงอย่างน้อยสามบรรทัด ถ้าคุณต้องการเติมลิสต์บ็อกซ์เข้าไปในแบบฟอร์ม ให้ใช้เครื่องมือสร้างลิสต์บ็อกซ์จากทูลบาร์เช่นที่แสดงในรูปที่ 5.16 (ลิสต์บ็อกซ์ แบบคิงลง เช่น เช็ดคิงบ็อกซ์ที่ปรากฏอยู่ในวินโดว์ Properties ของ Visual Basic เป็นคอมโบบ็อกซ์ประเภทหนึ่ง ซึ่งจะพิจารณาในตอนถัดไป)



รูปที่ 2.16

เครื่องมือสร้างลิสต์บ็อกซ์

คุณไม่สามารถนิยามเนื้อหาของลิสต์บ็อกซ์ในเวลาออกแบบ คุณจะต้องใช้กลวิธี *AddItem* เพื่อป้อนข้อมูลเข้าไปในลิสต์บ็อกซ์โดยผ่านทางโปรแกรมของคุณ ตามปกติจะเป็นการดีที่คุณกำหนดค่าเริ่มต้นให้แก่ลิสต์บ็อกซ์ไว้ในกระบวนการงานอีเวนต์ *Load* ของแบบฟอร์มที่ประกอบไปด้วยลิสต์บ็อกซ์นั้น

คุณสมบัติ

ลิสต์บ็อกซ์ประกอบไปด้วยคุณสมบัติตามปกตินั้นคือ *Enabled*, *FontBold*, *FontItalic*, *FontName*, *FontSize*, *FontUnderline*, *Height*, *Left*, *Name*, *Top*, *Visible* และ *Width* ส่วนคุณสมบัติพิเศษของลิสต์บ็อกซ์มีดังนี้

- Columns** เมื่อตั้งคุณสมบัตินี้ให้เป็น 0 (ซึ่งเป็นค่าโดยปริยาย) จะแสดงขึ้นรายการทั้งหมดออกมาในคอลัมน์เพียงคอลัมน์เดียว ถ้าตั้งคุณสมบัตินี้ให้เป็น 1 หรือมากกว่า จะจัดขึ้นรายการไว้ในเค้าโครงแบบมีหลายคอลัมน์ ค่าของคุณสมบัติที่ตั้งไว้จะบ่งบอกว่าในช่วงเวลาหนึ่ง ๆ นั้นจะเห็นคอลัมน์เป็นจำนวนเท่าใด ถ้าจำเป็นก็จะเดิมสกรอลบาร์ตามแนวนอนให้แก่ลิสต์บ็อกซ์
- List** คุณสมบัตินี้ซึ่งไม่สามารถดัดแปลงได้ในเวลาออกแบบ จะครอบครองอาร์เรย์ของค่าทั้งหมดที่เก็บอยู่ในลิสต์บ็อกซ์ คุณสามารถเข้าถึงขึ้นรายการแต่ละชั้นในลิสต์บ็อกซ์โดยใช้วากยสัมพันธ์ดังนี้

Object.List (index)

- ListCount** คุณสมบัติ *listCount* ซึ่งไม่สามารถดัดแปลงได้โดยตรง จะประกอบไปด้วยจำนวนของขึ้นรายการในลิสต์บ็อกซ์
- ListIndex** การตั้งคุณสมบัตินี้แสดงหมายเลขของขึ้นรายการที่เลือกล่าสุด ค่าดัชนีของขึ้นรายการแรกในลิสต์บ็อกซ์ คือ 0 ค่าดัชนีของขึ้นรายการถัดไปคือ 1 และเป็นเช่นนั้นต่อเนื่องไป ถ้าไม่ได้เลือกขึ้นรายการใดเลยจะตั้ง *ListIndex* ให้เป็น -1 ถ้าคุณตั้งคุณสมบัติ

ListIndex ผ่านทางโปรแกรม แถบสว่างในรายการจะเปลี่ยนแปลงตามค่าที่ตั้งไว้ คุณสามารถหาข้อความของชั้นรายการที่เลือกไว้โดยการอ่านคุณสมบัติ Text

MultiSelect คุณสมบัตินี้ควบคุมว่าจะสามารถเลือกชั้นรายการในลิสต์บ็อกซ์ได้เป็นจำนวนเท่าใด ถ้าตั้งคุณสมบัตินี้ให้เป็น 0-None นั้นหมายถึงในแต่ละครั้งผู้ใช้สามารถเลือกชั้นรายการได้เพียงหนึ่งชั้น (นั่นคือไม่ยอมให้มีการเลือกหลาย ๆ ชั้นรายการ) ถ้าตั้งให้เป็น 1-Simple ผู้ใช้จะสามารถเลือกชั้นรายการได้มากกว่าหนึ่งชั้น เมื่อตั้ง MultiSelect ให้เป็น 2-Extended ผู้ใช้สามารถเลือกชั้นรายการจำนวนหนึ่งโดยการกดปุ่มเมาส์บนชั้นรายการแรก กดเป็น Shift ค้างไว้ และกดปุ่มเมาส์บนชั้นรายการสุดท้ายที่อยู่ในพิสัย ผู้ใช้ยังสามารถกดแป้น Ctrl ค้างไว้ และกดปุ่มเมาส์บนชั้นรายการเพื่อเลือกหรือจัดการเลือกชั้นรายการแต่ละชั้น แม้แต่เมื่อเลือกชั้นรายการหลาย ๆ ชั้น คุณสมบัติ ListIndex และ Text จะแสดงเฉพาะการเลือกล่าสุด คุณจะต้องตรวจสอบแต่ละอ็อบเจกต์ของคุณสมบัติ Selected เพื่อหาว่าได้เลือกชั้นรายการชั้นใด

Selected คุณสมบัติ Selected คืออาร์เรย์ที่มีอ็อบเจกต์ถูกตั้งให้เป็น True หรือ False สำหรับชั้นรายการแต่ละชั้นในลิสต์บ็อกซ์ การตั้งให้เป็น True แสดงว่าเลือกชั้นรายการนั้น การตั้งให้เป็น False แสดงว่าไม่ได้เลือกชั้นรายการนั้น ในการตรวจสอบสถานะของชั้นรายการชั้นใดชั้นหนึ่ง ให้ระบุดัชนีของชั้นรายการนั้น โดยการใช้วากยสัมพันธ์ดังต่อไปนี้

`object.Selected (index)`

Sorted ถ้าคุณต้องการใช้ Visual Basic เก็บรักษาชั้นรายการ โดยให้เรียงตามลำดับอักษรคุณก็ควรจะต้องตั้งคุณสมบัตินี้ให้เป็น True เมื่อตั้งคุณสมบัตินี้ให้เป็น False (ค่าโดยปริยาย) ก็ will แสดงชั้นรายการออกมาโดยเรียงตามลำดับที่เดิมเข้าไป

Text คุณสมบัตินี้ซึ่งไม่สามารถดัดแปลงได้โดยตรง ประกอบไปด้วยข้อความของชั้นรายการที่เลือกไว้ล่าสุด

อีเวนต์

ลิสต์บ็อกซ์จะรับทั้งอีเวนต์ Click และ DbClick ตามปกติคุณไม่ควรจะลงรหัสกระบวนการงาน Click ทั้งนี้เพราะในขั้นแรกผู้ใช้อาจเลือกชั้นรายการชั้นหนึ่ง และหลังจากนั้นตกลงใจจะเลือกชั้นรายการที่ต่างไป โปรแกรมของคุณนั้นควรจะให้อ่านคุณสมบัติ Text ในเวลาที่ผู้ใช้กดปุ่มเมาส์บนปุ่มคำสั่ง หรือในเวลาที่เกิดอีเวนต์ DbClick ขึ้น

กระบวนการงานและกลวิธี

กลวิธีที่ใช้บ่อยที่สุดสำหรับลิสต์บ็อกซ์ประกอบไปด้วยกลวิธี *AddItem*, *Clear* และ *RemoveItem* ทั้งสามกลวิธีนี้ให้คุณสามารถดัดแปลงเนื้อหาของลิสต์บ็อกซ์ได้ในเวลาดำเนินงาน

AddItem กลวิธีนี้ให้คุณสามารถแทรกบรรทัดของข้อความเข้าไปไว้ในลิสต์บ็อกซ์ คุณระบุข้อความที่ต้องการแทรก และอาจมีดัชนีของชั้นรายการที่แทรก ถ้าคุณไม่ระบุดัชนี Visual Basic ก็จะแทรกชั้นรายการเอาไว้หลังชั้นรายการทั้งหมดที่มีอยู่ ถ้าตั้งคุณสมบัติ *Sorted* ให้เป็น *True* ไม่ต้องระบุค่าดัชนี ในกรณีนี้ Visual Basic จะแทรกชั้นรายการเรียงตามลำดับอักษร วากยสัมพันธ์ของกลวิธี *AddItem* จะเป็นดังนี้

```
object.AddItem text [, index]
```

Clear กลวิธีนี้จะกำจัดชั้นรายการทั้งหมดออกจากลิสต์บ็อกซ์ วากยสัมพันธ์สำหรับกลวิธี *Clear* ก่อนข้างจะง่าย ๆ คือ

```
object.Clear
```

RemoveItem กลวิธี *RemoveItem* ใช้ในการกำจัดข้อความบรรทัดหนึ่งออกจากลิสต์บ็อกซ์คุณต้องระบุดัชนีของชั้นรายการที่คุณต้องการกำจัด โดยการใช้วากยสัมพันธ์ดังต่อไปนี้

```
object.RemoveItem index
```

คอมโบบ็อกซ์

คอมโบบ็อกซ์สร้างขึ้นมาจากเครื่องมือตามที่แสดงอยู่ในรูปที่ 2.17 การที่ให้ชื่อคอมโบบ็อกซ์ก็เพราะว่าบ็อกซ์นี้สามารถรวมทั้งบ็อกซ์และลิสต์บ็อกซ์เอาไว้ในตัวควบคุมตัวเดียว อย่างไรก็ตามคอมโบบ็อกซ์จะแตกต่างจากลิสต์บ็อกซ์ตรงที่ไม่สนับสนุนการมีหลายคอลัมน์ คุณสามารถกำหนดให้คอมโบบ็อกซ์มีพฤติกรรมอย่างใดอย่างหนึ่งในสามชนิด โดยอาศัยการตั้งคุณสมบัติ *Style*



รูปที่ 2.17

เครื่องมือสร้างคอมโบบ็อกซ์

คุณสมบัติ

นอกเหนือจากคุณสมบัติตามปกติแล้ว คอมโบบ็อกซ์มีคุณสมบัติที่น่าสนใจอยู่สองคุณสมบัติ นั่นคือคุณสมบัติ Style และคุณสมบัติ Text

- Style** คุณสมบัติ Style ซึ่งกำหนดชนิดของคอมโบบ็อกซ์ และกำหนดว่าคอมโบบ็อกซ์มีพฤติกรรมเช่นใด สามารถตั้งให้เป็น 0,1 หรือ 2
- เมื่อตั้ง Style ให้เป็น 2 คอมโบบ็อกซ์จะกลายเป็นลิสต์บ็อกซ์แบบดิ่งลง ในตอนแรกเริ่มจะเห็นชั้นของรายการเพียงชั้นเดียว แต่เมื่อผู้ใช้กดปุ่มเมาส์บนปุ่มลูกศรชี้ลงที่อยู่ด้านข้างบ็อกซ์ รายการที่สมบูรณ์จะหย่อนลงมา ซึ่งแสดงชั้นรายการทั้งหมดและยอมให้ผู้ใช้สามารถเลือกได้ชั้นใดชั้นหนึ่งเช่นเดียวกันกับในลิสต์บ็อกซ์ ตัวอย่างของบ็อกซ์ประเภทนี้ที่คุ้นเคยคือ ออปเจกต์บ็อกซ์ที่อยู่ในวินโดว์ Properties ของ Visual Basic
- การตั้ง Style ให้เป็น 1 จะสร้างในสิ่งที่เรียกว่าเป็นคอมโบบ็อกซ์ง่าย ๆ ซึ่งเป็นบริเวณที่จะแก้ไขที่ผู้ใช้สามารถป้อนข้อความเข้าไปได้พร้อมกับมีลิสต์บ็อกซ์มาตรฐานอยู่ได้บริเวณนั้นโดยตรง รายการไม่ได้ถูกหย่อนลงมาแต่จะแสดงออกมาเสมอ ผู้ใช้อาจเลือกชั้นรายการชั้นใดชั้นหนึ่งจากรายการทั้งหมด หรือป้อนข้อความเข้าไปในบริเวณที่จะแก้ไข
- การตั้งคุณสมบัติ Style ให้เป็น 0 จะสร้างคอมโบบ็อกซ์แบบดิ่งลง ซึ่งมีลักษณะคล้ายกับลิสต์บ็อกซ์แบบดิ่งลง แต่ยังให้ผู้ใช้สามารถป้อนข้อความหรือเลือกจากรายการแบบดิ่งลง เขตดิ่งบ็อกซ์ในวินโดว์ Properties ของ Visual Basic จะทำหน้าที่ทำนองเดียวกันกับคอมโบบ็อกซ์แบบดิ่งลง
- Text** คุณสมบัตินี้สามารถประกอบไปด้วยข้อความของชั้นรายการที่เลือกจากรายการ หรือข้อความที่ผู้ใช้ป้อนเข้าไปในบริเวณที่จะแก้ไขโดยตรง

อีเวนต์

คอมโบบ็อกซ์จะตอบสนองต่ออีเวนต์อย่างไรนั้นขึ้นอยู่กับรูปแบบของการดำเนินการของคอมโบบ็อกซ์นั้น นั่นคือขึ้นอยู่กับว่าจะตั้งคุณสมบัติ Style ให้เป็นอย่างไร เช่น คอมโบบ็อกซ์จะรับอีเวนต์ DbClick ก็เฉพาะเมื่อได้ตั้งคุณสมบัติ Style ให้เป็น 1 (คอมโบบ็อกซ์อย่างง่าย) ในอีกสองรูปแบบนั้นจะรับเฉพาะอีเวนต์ Click เท่านั้น รูปแบบสองรูปแบบที่ยอมให้มีการป้อนข้อความเข้าไป

สามารถรับอีเวนต์ *Change* เมื่อผู้ใช้พิมพ์เข้าไปในบริเวณที่จะแก้ไขของคอมโบบ็อกซ์ โดยทั่วไปแล้ว คุณจะเพียงแค่ต้องการให้โปรแกรมของคุณสามารถอ่านคุณสมบัติ *Text* ของคอมโบบ็อกซ์หลังจากที่ผู้ใช้ได้ทำการเลือกแล้ว

กระบวนการและกลวิธี

กลวิธี *AddItem*, *Clear* และ *RemoveItem* ที่ได้บรรยายไว้สำหรับลิสต์บ็อกซ์สามารถนำมาใช้กับคอมโบบ็อกซ์ได้เช่นกัน และใช้ในการทำงานเดียวกัน

สกรอลล์บาร์ตามแนวนอนและตามแนวตั้ง

สกรอลล์บาร์ตามแนวนอนและตามแนวตั้ง (ตามที่แสดงไว้ในรูปที่ 5.22) จะประพฤติปฏิบัติในการทำงานเดียวกัน ยกเว้นแต่ทิศทางของแท่งจะแตกต่างกัน ออบเจกต์เหล่านี้เพียงแต่รายงานตำแหน่งของสกรอลล์บ็อกซ์ที่อยู่ในสกรอลล์บาร์ คุณจะเป็นคนคอยควบคุมช่วงของสกรอลล์บาร์ และการเพิ่มค่าที่สกรอลล์บ็อกซ์จะสามารถเลื่อนไปข้างหน้า เช่น ถ้าคุณกำลังใช้สกรอลล์บาร์แทนลิ้นบังคับหรือ *throttle* ของรถยนต์ ฉะนั้นแท่งอาจให้อยู่ในช่องตั้งแต่ 0 ถึง 100 พร้อมกับเพิ่มค่าขึ้นทีละ 5 ตัวควบคุมปริมาณอาจอยู่ในช่วงตั้งแต่ 0 ถึง 10 พร้อมกับเพิ่มค่าขึ้นทีละ 1



รูปที่ 2.18

สกรอลล์บาร์ตามแนวนอนและตามแนวตั้ง

ถ้าต้องการสร้างออบเจกต์เหล่านี้ให้ใช้เครื่องมือสร้างสกรอลล์บาร์ตามแนวนอน หรือเครื่องมือสร้างสกรอลล์บาร์ตามแนวตั้ง จากทูลบาร์บ็อกซ์ของ Visual Basic



รูปที่ 2.19

เครื่องมือสกรอลล์บาร์ตามแนวนอน



รูปที่ 2.20

เครื่องมือสร้างสกรอลล์บาร์ตามแนวตั้ง

คุณสมบัติ

คุณสมบัติมาตรฐานที่ปรากฏอยู่ในสกรอลล์บาร์ประกอบไปด้วย Enabled, Height, Left, Name, Top, Visible และ Width คุณสมบัติพิเศษของทั้งสกรอลล์บาร์ตามแนวนอนและตามแนวตั้งประกอบไปด้วย LargeChange, Max, Min, SmallChange และ Value

LargeChange การตั้งคุณสมบัตินี้แทนค่าที่จะเพิ่มเข้าไป หรือที่จะลบออกจากจำนวนที่บรรจุอยู่ในคุณสมบัติ Value เมื่อผู้ใช้คลิกปุ่มเมาส์ภายในสกรอลล์บาร์

Max คุณสมบัตินี้สามารถตั้งให้เป็นจำนวนที่อยู่ในช่วงตั้งแต่ -32,768 ตลอดจนถึง 32,767 เมื่อสกรอลล์บ็อกซ์อยู่ที่ตำแหน่งขวาสุดหรือล่างสุด คุณสมบัติ Value จะถูกตั้งให้เป็นจำนวนนี้

Min คุณสมบัตินี้สามารถตั้งให้เป็นจำนวนที่อยู่ในช่วง -32,768 ตลอดจนถึง 32,767 เมื่อสกรอลล์บ็อกซ์อยู่ที่ตำแหน่งซ้ายสุดหรือบนสุด คุณสมบัติ Value จะถูกตั้งให้เป็นจำนวนนี้

SmallChange การตั้งคุณสมบัตินี้แทนค่าที่จะเพิ่มเข้าไปหรือลบออกจากจำนวนที่บรรจุอยู่ในคุณสมบัติ Value เมื่อผู้ใช้คลิกปุ่มเมาส์บนลูกศรเลื่อนจอภาพที่อยู่ทางด้านใดด้านหนึ่งของสกรอลล์บาร์

Value คุณสมบัตินี้ประกอบไปด้วยจำนวนที่แทนตำแหน่งล่าสุดของสกรอลล์บ็อกซ์ที่อยู่ภายในสกรอลล์บาร์ ถ้าคุณตั้งคุณสมบัตินี้ผ่านทางโปรแกรม Visual Basic จะโยกย้ายสกรอลล์บ็อกซ์ให้อยู่ในตำแหน่งที่เหมาะสมการตั้งคุณสมบัติ Value ให้อยู่นอกช่องที่นิยามไว้ด้วยค่ากำหนดของคุณสมบัติ Max และ Min จะก่อให้เกิดข้อความระบุนความผิดพลาด

อีเวนต์

อีเวนต์พื้นฐานที่เกี่ยวข้องกับสกรอลล์บาร์คือ อีเวนต์ Scroll และ Change ในขณะที่มีการลากสกรอลล์บ็อกซ์ภายในสกรอลล์บาร์จะมีการดำเนินการอีเวนต์ Scroll ซ้ำ ๆ กัน ส่วนอีเวนต์ Change จะได้รับสัญญาณหลังจากที่ได้ตัดแปลงตำแหน่งของสกรอลล์บาร์ ฉะนั้นใช้อีเวนต์ Scroll ในการบันทึกการเปลี่ยนแปลงเชิงไดนามิกในสกรอลล์บาร์ และใช้อีเวนต์ Change เพื่อให้ได้ค่าสุดท้ายของสกรอลล์บาร์

กระบวนการและกลวิธี

กระบวนการและกลวิธีที่ได้บรรยายไว้ก่อนหน้านี้ไม่สามารถนำมาใช้กับสกรอลล์บาร์

ตัวจับเวลา (Timer)

ตัวจับเวลาซึ่งสร้างขึ้นโดยอาศัยเครื่องมือสร้างตัวจับเวลาตามที่แสดงอยู่ในรูปที่ 2.21 เป็นออบเจกต์ที่สามารถทำให้อีเวนต์ขึ้นด้วยช่วงที่สม่ำเสมอ คุณเขียนโปรแกรมตัวจับเวลาด้วยการตั้งคุณสมบัติ Interval ให้เป็นจำนวนของมิลลิวินาที (1/1000 วินาที) ที่ควรจะสูญเสียไประหว่างอีเวนต์แต่ละอีเวนต์ของตัวจับเวลา



รูปที่ 2.21

เครื่องมือสร้างตัวจับเวลา

ตัวจับเวลาจะถูกจำกัดด้วยความสามารถของฮาร์ดแวร์ ระบบคอมพิวเตอร์ส่วนบุคคลส่วนใหญ่จะสนับสนุนอำนาจจำแนกของตัวจับเวลาไม่เกินกว่า 18 อีเวนต์ต่อวินาที หรือประมาณ 56/1000 วินาทีระหว่างอีเวนต์ ในทางปฏิบัติอำนาจจำแนกนี้หมายถึงว่า ค่าที่น้อยกว่า 56 จะไม่เป็นประโยชน์เลข หรือกล่าวอีกนัยหนึ่ง ค่าช่วงเวลาที่แตกต่างกัน 5, 10 หรือแม้แต่ 20 จะสร้างผลลัพธ์ที่เหมือนกัน ข้อจำกัดนี้ไม่ได้หมายถึงว่าคุณควรจะคำนวณอย่างสะเพร่า แต่หมายถึงว่าคุณไม่ควรจะไว้วางใจตัวจับเวลาว่ามีความแม่นยำอย่างไม่น่าเชื่อ

นอกจากนี้อีเวนต์ของตัวจับเวลาบางอีเวนต์อาจสูญเสียลำดับการทำงาน สมมติว่าโปรแกรมของคุณได้รับอีเวนต์ของตัวจับเวลาวินาทีละหนึ่งครั้ง แต่ส่วนประกอบนั้นของโปรแกรมเกี่ยวข้องกับ การคำนวณที่ซับซ้อนมากที่ต้องใช้เวลา 10 วินาทีเพื่อคำนวณ เมื่อการคำนวณนั้นเสร็จสิ้นแล้ว คุณอาจคาดหมายว่าอีเวนต์ของตัวจับเวลา 10 อีเวนต์อาจกองซ้อนกันขึ้นไปด้วยวิธีการใดวิธีหนึ่ง แต่ตามความเป็นจริงแล้วมีอีเวนต์ของตัวจับเวลาเพียง 1 อีเวนต์ที่จะเกิดขึ้นทันทีหลังการคำนวณ ก่อนที่จะก่อกำเนิดอีเวนต์ของตัวจับเวลาขึ้นนั้น ระบบจะคอยตรวจสอบก่อนว่าอีเวนต์ของตัวจับเวลาก่อนหน้านี้อยู่ค้างอยู่หรือไม่ ถ้ายังค้างอยู่ก็จะไม่ก่อกำเนิดอีเวนต์ของตัวจับเวลาขึ้น การที่มีข้อจำกัดเช่นนี้ อาจทำให้คุณสงสัยถึงการใช้ประโยชน์จากตัวจับเวลา แต่จริง ๆ แล้ว ตัวจับเวลา มีประโยชน์มากทีเดียวแม้ว่าจะไม่ใช่แหล่งของการจับเวลาที่มีความเที่ยงตรง เช่นอาจใช้ตัวจับเวลาคอย ดับโปรแกรมของคุณด้วยช่วงเวลาที่เหมาะสม

คุณสมบัติ (Property)

คุณสมบัติ Name และ Enabled เป็นคุณสมบัติมาตรฐานของตัวจับเวลา คุณสมบัติที่สำคัญที่สุดของออบเจกต์ตัวจับเวลาคือ คุณสมบัติ Interval คุณตั้งคุณสมบัตินี้ให้เป็นจำนวนของมิลลิวินาทีที่อยู่ระหว่างอีเวนต์ของตัวจับเวลา พร้อมกับมีค่าอยู่ในช่วงตั้งแต่ 0 ตลอดจนถึง 65,535 ซึ่งในทางทฤษฎี จะก่อกำเนิดอีเวนต์ด้วยอัตราที่อยู่ในช่วงตั้งแต่วินาทีละ 1000 อีเวนต์ไปจนถึงน้อยกว่า 1 อีเวนต์ทุก ๆ หนึ่งนาาที ถ้าตั้งช่วงเวลาเป็น 0 จะเทียบเท่ากับการทำตัวจับเวลาให้ใช้การไม่ได้ ถ้าคุณต้องการก่อกำเนิดอีเวนต์ n ครั้งต่อวินาที ให้ใช้สูตร $1000/n$ เพื่อหาการตั้งค่าสำหรับคุณสมบัติ Interval เช่นถ้าคุณต้องการให้อีเวนต์ของตัวจับเวลาเกิดขึ้นทุก ๆ ครึ่งวินาที (สองครั้งต่อวินาที) การคำนวณ $1000/2$ แสดงว่าควรตั้งค่า Interval ให้เป็น 500

อีเวนต์

ไม่มีกระบวนการหรือกลวิธีดำเนินการบนออบเจกต์ตัวจับเวลา

เส้นและรูปทรง

ถ้าต้องการเพิ่มเติมลักษณะปรากฏของแบบฟอร์มในแอปพลิเคชัน คุณสามารถเพิ่มเติมเส้นง่าย ๆ และกราฟิกรูปทรงทึบโดยการใช้เครื่องมือสร้างเส้น และเครื่องมือสร้างรูปทรงของทูลบ็อกซ์ ซึ่งได้แสดงเอาไว้ในรูปที่ 2.22 และ 2.23 ออบเจกต์ที่สร้างขึ้นด้วยเครื่องมือเหล่านี้จะปรากฏอยู่ในชั้นกราฟิกเดียวกับออบเจกต์ภาพลักษณ์



รูปที่ 2.22

เครื่องมือสร้างเส้น



รูปที่ 2.23

เครื่องมือสร้างรูปทรง

ตัวควบคุมเส้นและรูปทรง (รวมทั้งตัวควบคุมภาพลักษณ์) มีประโยชน์สำหรับทำให้แบบฟอร์มประกอบไปด้วยภาวะส่วนหลังที่มองเห็นได้ เครื่องมือสร้างเส้นสร้างเซกเมนต์เส้นตรงง่าย ๆ ขึ้น คุณสามารถเปลี่ยนขนาด สี และแบบแผนของเส้นด้วยการดัดแปลงคุณสมบัติของออบเจกต์นั้น เครื่องมือสร้างรูปทรงจะสร้างสี่เหลี่ยมผืนผ้าบนจอภาพเสมอ แต่โดยอาศัยการดัดคุณสมบัติ Shape ของออบเจกต์ จะทำให้คุณสามารถสร้างวงกลม รูปกลมรี และสี่เหลี่ยมทรงมน นอกจากนี้คุณสามารถดัดสี และแบบแผนของรูปทรง

คุณสมบัติ

ออบเจกต์ทั้งสองตัวนี้มีคุณสมบัติ Name และ Visible ที่พบปกติ ออบเจกต์รูปทรงยังประกอบไปด้วยคุณสมบัติมาตรฐานคือ Height, Left, Top และ Width ออบเจกต์เส้นมีคุณสมบัติกำหนด

ตำแหน่ง $x1$, $y1$ และ $x2$, $y2$ ซึ่งเป็นพิกัดของจุดขุดติสองจุดของเส้น คุณสมบัติอื่น ๆ จะกำหนดว่า
 ออบเจ็กต์ปรากฏขึ้นบนจอภาพอย่างไร

BorderColor คุณสมบัตินี้จะกำหนดสีของเค้าโครงของออบเจ็กต์ (หรือในกรณีของออบเจ็กต์เส้น
 เป็นสีของเส้น) คุณตั้งคุณสมบัติ **BorderColor** โดยการเลือกสีจากชุดอุปกรณ์ใน
 ทำนองเดียวกันกับที่คุณตั้งคุณสมบัติ **BackColor** และ **ForeColor** ของแบบฟอร์ม

BorderStyle คุณสมบัติ **BorderStyle** ควบคุมว่าจะวาดเส้นหรือเค้าโครงของรูปทรงอย่างไรคุณ
 สามารถใช้ค่ากำหนดได้อย่างใดอย่างหนึ่งในเจ็ดอย่างดังนี้

0 - Transparent

1 - Solid

2 - Dash

3 - Dot

4 - Dash-Dot

5 - Dash-Dot-Dot

6 - Inside Solid

BorderWidth คุณสมบัตินี้ให้คุณสามารถระบุความกว้างของเส้นหรือเค้าโครงของรูปทรง ถ้าคุณตั้ง
BorderWidth ให้เป็นค่าอื่น ๆ ที่นอกเหนือจาก 1 คุณไม่สามารถใช้ค่าแบบจุดหรือ
 แบบเส้นประของ **BorderStyle**

ออบเจ็กต์รูปทรงยังมีคุณสมบัติ **FillColor** และ **FillStyle**

FillColor คุณสมบัติ **FillColor** คล้ายกับคุณสมบัติ **BorderColor** คือให้คุณสามารถนิยามสี ภาย
 ในของรูปทรง คุณตั้งคุณสมบัตินี้ด้วยแนวทางเดียวกันกับที่คุณตั้งคุณสมบัติ
BorderColor นั่นคือโดยการใช้ชุดอุปกรณ์สี

FillStyle การตั้งคุณสมบัติ **FillStyle** จะกำหนดแบบแผนภายในของออบเจ็กต์รูปทรง

0 - Solid

1 - Transparent

2 - Horizontal Line

3 - Vertical Line

4 - Upward Diagonal

5 - Downward Diagonal

6 - Cross

7 - Diagonal Cross

อีเวนต์

ออบเจกต์เส้นและออบเจกต์รูปทรงไม่รับอีเวนต์ใด ๆ

โครฟลิสต์บ็อกซ์ ไคเรกทอรีลิสต์บ็อกซ์ และไฟล์ลิสต์บ็อกซ์

ทูลบ็อกซ์ของ Visual Basic ประกอบไปด้วยเครื่องมือหลายประเภทตามที่แสดงในรูปที่ 2.24, 2.25 และ 2.26 ซึ่งให้คุณสามารถสร้างไคเรกทอรีลิสต์บ็อกซ์ตามข้อกำหนดที่อินเตอร์เฟซเข้ากับระบบไฟล์ ตัวควบคุมระบบไฟล์แต่ละตัวจะจัดการส่วนประกอบที่แยกกันของระบบ ไครฟลิสต์บ็อกซ์จะปรากฏเป็นลิสต์บ็อกซ์ที่แสดงไคเรกทอรีย่อยทั้งหมดที่อยู่ในไคเรกทอรีล่าสุดและไฟล์ลิสต์บ็อกซ์ที่แสดงไฟล์บางส่วนหรือทั้งหมดในไคเรกทอรีล่าสุด ด้วยเหตุผลส่วนประกอบเหล่านี้จะต้องทำงานด้วยกันทั้งหมด ดังนั้นการเขียนโปรแกรมที่จำเป็นต้องใช้กับตัวควบคุมระบบไฟล์จึงมีความซับซ้อนมากกว่าการเขียนโปรแกรมที่ใช้กับตัวควบคุมอื่น ๆ



รูปที่ 2.24

เครื่องมือสร้างโครฟลิสต์บ็อกซ์



รูปที่ 2.25

เครื่องมือสร้างไคเรกทอรีลิสต์บ็อกซ์



รูปที่ 2.26

เครื่องมือสร้างไฟล์ลิสต์บ็อกซ์

คุณสมบัติ

ตัวควบคุมระบบไฟล์สามตัวมีคุณสมบัติอยู่เป็นจำนวนมากคือ Enabled, FontBold, FontItalic, FontName, FontSize, Height, Left, Name, Top, Visible และ Width นอกจากนี้คุณสมบัตินี้ต่อไปนี้ก็มีความสำคัญ

- Drive** คุณสมบัตินี้ซึ่งใช้ได้เฉพาะกับไดรฟ์ลิสต์บ็อกซ์ ประกอบไปด้วยชื่อของไดรฟ์ที่เลือกไว้ล่าสุด
- FileName** เฉพาะไฟล์ลิสต์บ็อกซ์เท่านั้นที่มีคุณสมบัตินี้ FileName ซึ่งประกอบไปด้วยชื่อไฟล์ที่เลือกไว้ล่าสุดในไฟล์ลิสต์บ็อกซ์
- List, ListIndex** คุณสมบัตินี้ซึ่งใช้ได้เฉพาะกับไฟล์ลิสต์บ็อกซ์เท่านั้น และเหมือนกันกับคุณสมบัตินี้ List และ ListIndex ของลิสต์บ็อกซ์
- Path** ในไดเรกทอรีลิสต์บ็อกซ์และไฟล์ลิสต์บ็อกซ์ คุณสมบัตินี้ประกอบไปด้วยพาทล่าสุด คุณสมบัตินี้ Path ใช้ไม่ได้กับไดรฟ์ลิสต์บ็อกซ์
- Pattern** คุณสมบัตินี้ของไฟล์ลิสต์บ็อกซ์ ประกอบไปด้วยสตริงที่กำหนดว่าจะแสดงไฟล์ใดออกมา คุณสมบัตินี้สนับสนุนอักขระ * และ ? ในชื่อไฟล์เช่น การตั้งคุณสมบัตินี้ Pattern ให้เป็น *.DAT จะแสดงไฟล์ทั้งหมดที่มีส่วนขยายชื่อไฟล์ DAT ออกมา

อีเวนต์

ไดเรกทอรีลิสต์บ็อกซ์และไฟล์ลิสต์บ็อกซ์ตอบสนองต่ออีเวนต์ Click แม้ว่าไดรฟ์ลิสต์บ็อกซ์จะไม่ตอบสนองก็ตาม อีเวนต์ DbClick จะใช้ได้เฉพาะกับไฟล์ลิสต์บ็อกซ์เท่านั้น อีเวนต์อื่น ๆ อีกสองอีเวนต์มีความสำคัญต่อตัวควบคุมระบบไฟล์

- Change** อีเวนต์นี้รับรู้ได้เฉพาะโดยไดรฟ์และไดเรกทอรีลิสต์บ็อกซ์ อีเวนต์นี้จะเกิดขึ้นเมื่อใดก็ตามที่ผู้ใช้หรือโปรแกรมเปลี่ยนแปลงการเลือกในลิสต์บ็อกซ์
- PathChange** อีเวนต์นี้ใช้ได้เฉพาะกับไฟล์ลิสต์บ็อกซ์เท่านั้น อีเวนต์นี้จะเกิดขึ้นเมื่อใดก็ตามที่เปลี่ยนแปลงพาท ไม่ว่าจะเป็นเพราะการที่ผู้ใช้กดปุ่มเมาส์สองครั้งบนไดเรกทอรีหรือเพราะว่าโปรแกรมเปลี่ยนแปลงคุณสมบัตินี้ FileName หรือ Path อีเวนต์นี้ทำให้โปรแกรมของคุณสามารถปรับสิ่งที่บรรจุอยู่ในไดเรกทอรีลิสต์บ็อกซ์ที่เข้าคู่กัน

กระบวนการงานและกลวิธี

กระบวนการงานและกลวิธีเท่าที่พิจารณามานั้นไม่สามารถนำมาใช้กับตัวควบคุมระบบไฟล์ได้เลย

:

เครื่องมืออื่น ๆ ของ Visual Basic

ทูลบ็อกซ์ของ Visual Basic ยังประกอบไปด้วยเครื่องมืออื่น ๆ อีกจำนวนเล็กน้อย ซึ่งจะบรรยายไว้โดยย่อในที่นี้ ในหนังสือเล่มนี้จะได้กล่าวถึงเครื่องมือเหล่านี้ และอธิบายเจ็ดที่เครื่องมือเหล่านี้สร้างขึ้นอย่างโดยละเอียด ทั้งนี้เพราะต้องอาศัยความพยายามในการเขียนโปรแกรมค่อนข้างมาก และต้องอาศัยเทคนิคระดับสูง หลังจากที่คุณอ่านหนังสือเล่มนี้จบแล้วซึ่งจะมุ่งเน้นเกี่ยวกับพื้นฐาน คุณก็จะได้รับการเตรียมเพื่อให้อัดการกับความซับซ้อนของเครื่องมือเหล่านี้

กริด

เครื่องมือสร้างกริดซึ่งได้ลงภาพไว้ในรูปที่ 2.27 นำมาใช้สร้างตัวควบคุมกริดขึ้น กริดเป็นบล็อกของเซลล์สองมิติที่คล้ายคลึงกับแถวและคอลัมน์ของสเปรดชีต เราสามารถใช้ตัวควบคุมกริดเพื่อทำให้เกิดฟังก์ชันคล้ายสเปรดชีตขึ้นใน Visual Basic โครงการที่ใช้กริดจะต้องรวมเอาไฟล์ GRID.VBX เข้าไว้ในไฟล์โครงการด้วย



รูปที่ 2.27

เครื่องมือสร้างกริด

ข้อมูล

เครื่องมือสร้างข้อมูลตามที่แสดงอยู่ในรูปที่ 2.28 ใช้ในการสร้างตัวควบคุมข้อมูล เราอาศัยตัวควบคุมนี้ในการเข้าถึงข้อมูลจำเพาะในฐานข้อมูล ตัวควบคุมข้อมูลมีคุณสมบัติอยู่จำนวนหนึ่งที่นิยามการติดต่อระหว่างแอปพลิเคชันของ Visual Basic และฐานข้อมูลที่มุ่งหมายจะเข้าถึงก่อนใช้เครื่องมือสร้างข้อมูลใน Visual Basic จำเป็นต้องทำการแบ่งสรรไฟล์ให้อยู่ในสภาพใช้การ



รูปที่ 2.28

เครื่องมือสร้างข้อมูล

ไฟล์ AUTOLOAD.MAK

Visual Basic จะมีการรวมเอาไฟล์ GRID.VBX ไฟล์ CMDIALOG.VBX และไฟล์ MSOLE2.VBX เข้าไว้ในโครงการที่สร้างขึ้นอย่างโดยอัตโนมัติ อย่างไรก็ตาม ถ้าในโครงการไม่ได้ใช้ตัวควบคุมกริด โคอะลิกบ็อกซ์ร่วม หรือ OLE ก็สามารถลบไฟล์เหล่านี้ออกจากไฟล์โครงการ ในการป้องกัน Visual Basic ไม่ให้รวมเอาไฟล์เหล่านี้เข้าไว้ในโครงการอย่างโดยอัตโนมัติ เลือกคำสั่ง Open Project จากเมนู File และเปิดไฟล์ AUTOLOAD.MAK จากโคอะลิกบ็อกซ์ Open Project จากแต่ละครั้งที่สร้างโครงการใหม่ขึ้นมานั้น Visual Basic จะคัดลอกเนื้อหาของไฟล์ AUTOLOAD.MAK เข้าไว้ในโครงการใหม่ ถ้าลบใหม่ ถ้ลบไฟล์ GRID.VBX และไฟล์ MSOLE2.VBX ออกจาก AUTOLOAD.MAK (เลือก Remove File จากเมนู File) ก็จะไม่มีการคัดลอกไฟล์ทั้งสองนี้ปรากฏอยู่ในโครงการใหม่อีกต่อไป ให้คงเหลือไฟล์ CMDIALOG.VBX เอาไว้ในไฟล์ AUTOLOAD.MAK ในตอนนี้เครื่องมือสร้างกริด และเครื่องมือสร้าง OLE จะหายไปจากชุดบ็อกซ์ด้วย อย่างไรก็ตาม สามารถรวมเอาไฟล์และเครื่องมือเหล่านี้กลับเข้าไปใหม่โดยการใช้คำสั่ง Add File จากเมนู File

คอมมอนโคอะลิกบ็อกซ์

ไมโครซอฟต์ได้สร้างชุดคอมมอนโคอะลิกบ็อกซ์สำหรับฟังก์ชันมาตรฐานของ Windows เพื่อให้ผู้ใช้มีอินเตอร์เฟซแบบมาตรฐานสำหรับการดำเนินการร่วมในแอปพลิเคชันที่ดำเนินงานบน Windows รหัสสำหรับฟังก์ชันเหล่านี้จัดจำหน่ายเป็นครั้งแรกมาพร้อมกับ Windows 3.1 แต่สามารถนำมาใช้กับ Windows 3.0 ถ้าเติมไฟล์ COMMDLG.DLL เข้าไปในไดเรกทอรี SYSTEM Visual Basic ให้แนวทางที่จะเข้าถึงฟังก์ชันเหล่านี้โดยผ่านเครื่องมือสร้างคอมมอนโคอะลิกบ็อกซ์ การใช้เครื่องมือสร้างคอมมอนโคอะลิกบ็อกซ์ทำให้มั่นใจว่าแอปพลิเคชันที่สร้างขึ้นจะสอดคล้องกันกับรูปแบบมาตรฐานของการดำเนินการที่สนับสนุน

เครื่องมือสร้างคอมมอนไดอะล็อกตามที่แสดงในรูปที่ 2.29 มีลักษณะคล้ายกับเครื่องมือสร้างตัวจับเวลา กล่าวคืออยู่ในสภาพที่ไม่สามารถเห็นได้ในเวลาดำเนินงาน อย่างไรก็ตาม เมื่อบรรจุไอคอนไว้บนแบบฟอร์มแบบใดตัวแบบหนึ่งแล้ว จะสามารถก่อให้เกิดหนึ่งในการทำหน้าที่มาตรฐานห้าอย่างที่เครื่องมือนี้สนับสนุนอยู่ การทำหน้าที่ที่ทว่านี้คือ ไดอะล็อกบ็อกซ์มาตรฐานของ Windows เพื่อใช้กับ Open, Save As, Color, Font และ Printer นอกจากนี้เครื่องมือสร้างคอมมอนไดอะล็อกยังสามารถก่อให้เกิดระบบความช่วยเหลือของ Windows การที่จะสามารถเข้าถึงเครื่องมือสร้างคอมมอนไดอะล็อกได้นั้น จะต้องบรรจุไฟล์ COMMDLG.VBX เข้าไปในวินโดว์ Project



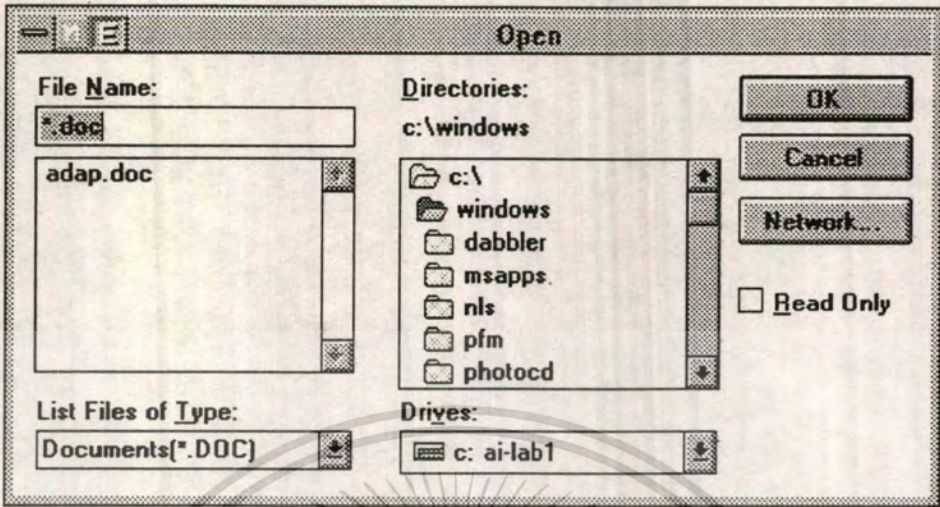
รูปที่ 2.29

เครื่องมือสร้างคอมมอนไดอะล็อก

การเอ็กซ์ิตฟังก์ชันที่เลือกไว้เริ่มต้นขึ้นเมื่อได้ตั้งคุณสมบัติ Action ของตัวควบคุมคอมมอนไดอะล็อก ข้อความสั่งของ Visual Basic จะหยุดชะงักจนกว่าจะบอกเลิกไดอะล็อกบ็อกซ์หลังจากนั้นจะสามารถตรวจสอบคุณสมบัติของตัวควบคุมเพื่อหาว่ามีกิจกรรมอะไรเกิดขึ้นไม่มีอีเวนต์หรือกลวิธีที่เกี่ยวข้องกับตัวควบคุมคอมมอนไดอะล็อก เรากำหนดการดำเนินการของตัวควบคุมโดยการตั้งค่าคุณสมบัติ และหลังจากนั้นกำหนดค่าให้แก่คุณสมบัติ Action นอกเหนือจากคุณสมบัติ Action แล้วฟังก์ชันทั้งหมดของคอมมอนไดอะล็อกจะใช้คุณสมบัติ CancelError ด้วย ค่าโดยปริยายของ CancelError คือ False แต่ถ้าตั้งให้เป็น True Visual Basic จะบ่งบอกความผิดพลาดออกมาถ้าผู้ใช้เลือกปุ่ม Cancel ในไดอะล็อกบ็อกซ์ ถ้ามีแผนจะตั้ง CancelError ให้เป็น True ควรจะใช้คุณลักษณะจัดการความผิดพลาดของ Visual Basic

Open

ไดอะล็อกบ็อกซ์ Open ตามที่แสดงในรูปที่ 2.30 จะเกิดขึ้นเมื่อกำหนดค่า 1 ให้แก่คุณสมบัติ Action เราสามารถตั้งคุณสมบัติดังต่อไปนี้เพื่อควบคุมลักษณะของไดอะล็อกบ็อกซ์ที่ปรากฏออกมาควรทราบไว้ด้วยว่าไดอะล็อกบ็อกซ์นี้ไม่ได้เปิดไฟล์จริง ๆ แต่เป็นการให้ผู้ใช้สามารถเลือกชื่อไฟล์ได้เท่านั้น



รูปที่ 2.30
ไอคอนบ็อกซ์ Open

คุณสมบัติ

DialogTitle คุณสมบัตินี้สามารถตั้งให้เป็นสตริงใด ๆ สตริงจะแสดงอยู่ในไตเติลบาร์ของไอคอนบ็อกซ์โดยปริยายแล้วชื่อเรียกจะเป็น “Open”

FileName ใช้คุณสมบัตินี้ตั้งชื่อแรกเริ่มในเท็กซ์บ็อกซ์ File Name ของไอคอนบ็อกซ์หลังจากที่บอกลึกไอคอนบ็อกซ์แล้ว อ่านคุณสมบัตินี้เพื่อหาชื่อของไฟล์ที่เลือกไว้

Filter เราสามารถตั้งคุณสมบัตินี้เพื่อให้จำกัดชื่อไฟล์ที่จะปรากฏอยู่ในไฟล์ลิสต์บ็อกซ์ คุณสมบัติ Filter ต้องเป็นสตริงข้อความที่มีคู่ส่วนประกอบหนึ่งคู่หรือมากกว่า คู่ส่วนประกอบ ประกอบไปด้วยคำบรรยายและอักขระวิลด์การ์ด แยกออกจากกันด้วยสัญลักษณ์ไปป์ (|) ถ้ามีคู่ส่วนประกอบหลาย ๆ คู่ ให้แยกออกจากกันด้วยสัญลักษณ์ไปป์เช่นเดียวกัน เช่น สตริงดังต่อไปนี้กำหนดคู่ส่วนประกอบสามคู่

“Documents (*.DOC) | *.DOC | Text Files (*.TXT) | *.txt | All Files *.*”

ฟิเตอร์แรกจะจำกัดรายการไฟล์ให้เป็นไฟล์ที่มีส่วนขยาย DOC ฟิเตอร์ที่สองใช้กับไฟล์ที่มีส่วนขยาย TXT และฟิเตอร์ที่สามยอมให้มีการเลือกไฟล์โดยไม่เจาะจงสังเกตดูจะเห็นว่าไม่มีช่องว่างก่อนและหลังอักขระไปป์

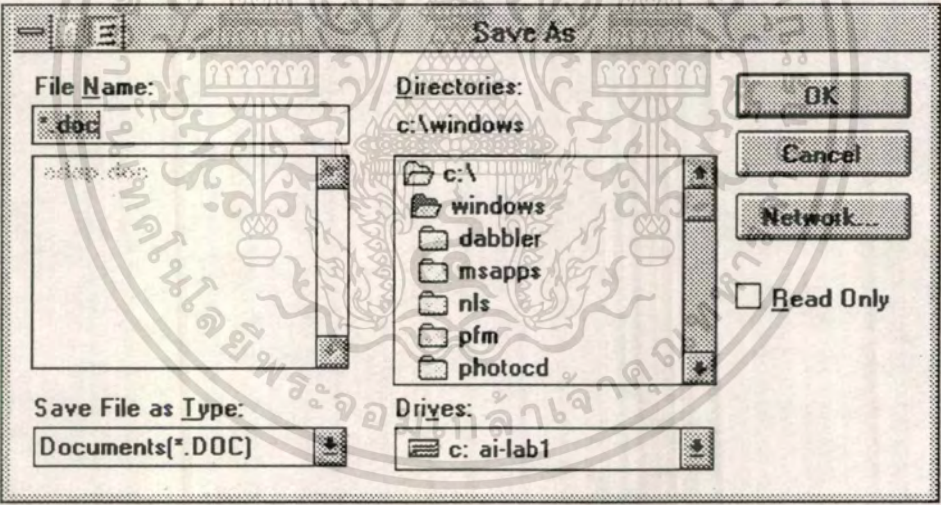
- FiltrIndex

-ตั้งคุณสมบัตินี้ให้เป็นจำนวนเต็มเพื่อแสดงว่าคู่ส่วนประกอบฟิเตอร์คู่ใดที่จะเป็นฟิเตอร์โดยปริยาย
- Flags

คุณสมบัติ Flag สามารถตั้งให้เป็นค่าหนึ่งค่าหรือมากกว่ารวมกันที่ควบคุมคุณลักษณะพิเศษของไดอะล็อกบ็อกซ์
ค่าตัวบ่งชี้ที่ถูกต้องได้ลงไว้ในไฟล์ CONSTANT.TXT และได้อธิบายอยู่ในระบบความช่วยเหลือแบบเชื่อมต่อตรงของ Visual Basic ตัวอย่างคือตัวบ่งชี้ OFN__FILEMUSTEXIST ซึ่งผลักดันให้ผู้เลือกใช้ไฟล์ที่มีอยู่

Save As

ไดอะล็อกบ็อกซ์ Save As ตามที่แสดงในรูปที่ 2.31 เกิดขึ้นเมื่อได้กำหนดค่า 2 ให้แก่คุณสมบัติ Action คุณสมบัติสำหรับ Save As จะเหมือนกันกับที่บรรยายข้างต้นสำหรับ Open ยกเว้นแต่ค่าโดยปริยายสำหรับคุณสมบัติ DialogTitle คือ Save As



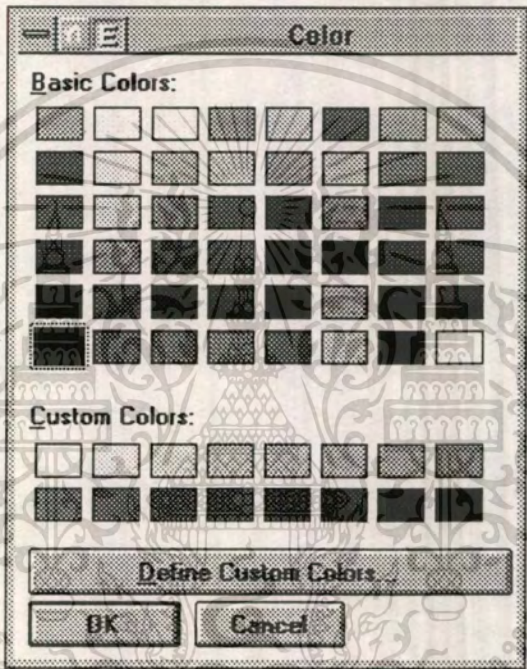
รูปที่ 2.31
ไดอะล็อกบ็อกซ์ Save As

คุณสมบัติ

DefaultExt คุณสมบัตินี้สามารถตั้งให้เป็นสตริง 1 ถึง 3 อักขระ ซึ่งนำไปใช้เป็นส่วนขยายโดยปริยายในเวลาที่กำหนดชื่อไฟล์โดยไม่มีส่วนขยาย

Color

ไดอะล็อกบ็อกซ์ Color ตามที่แสดงในรูปที่ 2.32 จะเกิดขึ้นเมื่อกำหนดค่า 3 ให้แก่คุณสมบัติ Action เราสามารถตั้งคุณสมบัติต่อไปนี้เพื่อควบคุมลักษณะของไดอะล็อกบ็อกซ์ที่ปรากฏออกมา



รูปที่ 2.32
ไดอะล็อกบ็อกซ์ Color

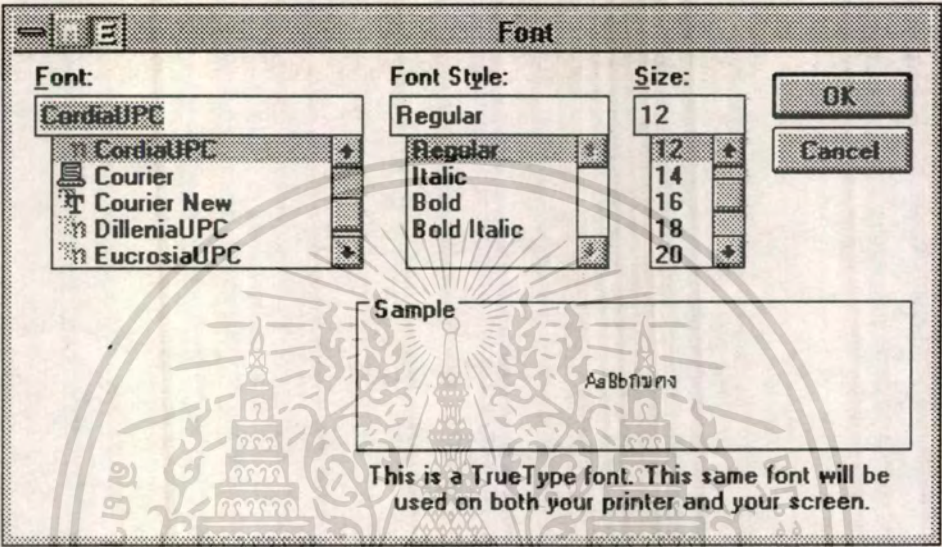
คุณสมบัติ

Color เราใช้คุณสมบัตินี้ตั้งสีแรกเริ่ม หลังจากบอกลีกไดอะล็อกบ็อกซ์แล้วสามารถอ่านคุณสมบัตินี้เพื่อหาสีที่ผู้ใช้เลือก

Flags ต้องตั้งคุณสมบัตินี้ให้เป็น CC__RGBINIT เพื่อตั้งหรืออ่านคุณสมบัติ Color

Font

ไดอะล็อกบ็อกซ์ Font ตามที่แสดงในรูปที่ 2.33 เกิดขึ้นเมื่อกำหนดค่า 4 ให้แก่คุณสมบัติ Action เราสามารถตั้งคุณสมบัติใด ๆ ดังต่อไปนี้เพื่อควบคุมลักษณะของไดอะล็อกบ็อกซ์ที่ปรากฏออกมา



รูปที่ 2.33

ไดอะล็อกบ็อกซ์ Font

คุณสมบัติ

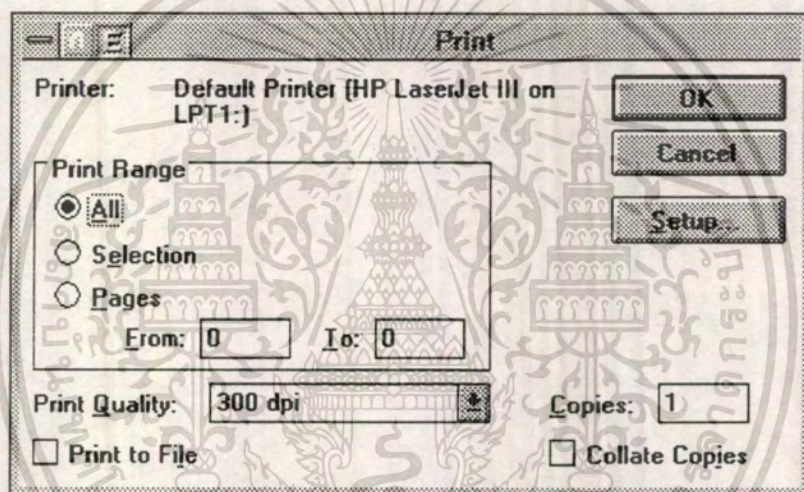
- Color** คุณสมบัตินี้ทำให้สามารถตั้งหรืออ่านสีของฟอนต์ได้ (จะอยู่ในสภาพพร้อมใช้งาน เฉพาะในเวลาที่คุณสมบัติ Flags ได้รวมเอาตัวบ่งชี้ CF_EFFECTS ไว้ด้วย)
- Flags** เราสามารถตั้งคุณสมบัตินี้เพื่อคัดแปลงพฤติกรรมของไดอะล็อกบ็อกซ์ อย่างไรก็ตาม คุณสมบัตินี้จะต้องมีค่าเหล่านี้ค่าใดค่าหนึ่งไว้ CF__PRINTERFONTS, CF__SCREENFONTS หรือ CF__BOTH ค่าเหล่านี้จะกำหนดว่าให้ส่งฟอนต์ใดใน ไดอะล็อกบ็อกซ์ (เฉพาะฟอนต์เครื่องพิมพ์ เฉพาะฟอนต์จอภาพ หรือทั้งสองชุด)

FontName เราสามารถตั้งคุณสมบัตินี้เพื่อกำหนดค่าเริ่มต้นฟอนต์โดยปริยาย และอ่านเพื่อหาการเลือกฟอนต์ของผู้ใช้

FontSize เราสามารถตั้งหรืออ่านคุณสมบัตินี้ และหาขนาดพอยต์ของฟอนต์

Print

ไคอะล็อกบ็อกซ์ Print ตามที่แสดงในรูปที่ 2.34 เกิดขึ้นเมื่อกำหนดค่า 5 ให้แก่คุณสมบัติ Action เราสามารถตั้งคุณสมบัติต่อไปนี้เพื่อควบคุมลักษณะของไคอะล็อกบ็อกซ์ที่ปรากฏออกมา



รูปที่ 2.34

ไคอะล็อกบ็อกซ์ Print

คุณสมบัติ

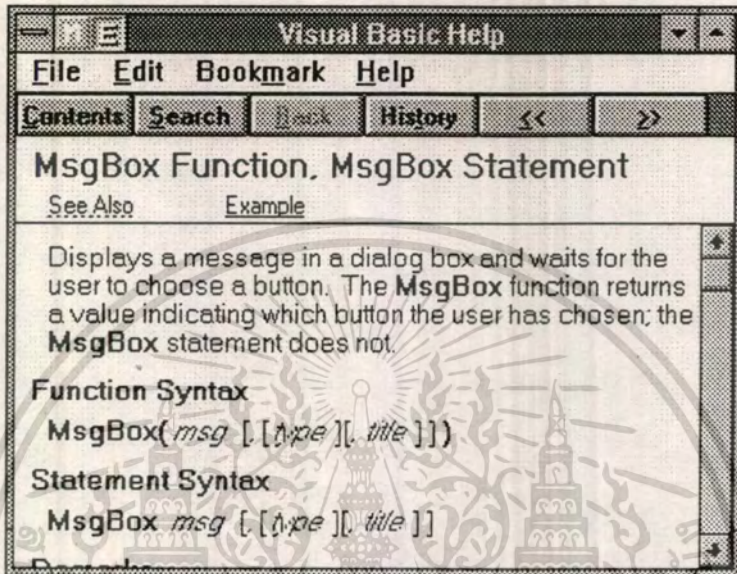
Copies คุณสมบัติจำนวนเต็มนี้กำหนดจำนวนของสำเนาที่จะพิมพ์ออกมา

Flags คุณสมบัตินี้จัดการกับคุณลักษณะพิเศษของไคอะล็อกบ็อกซ์ เราสามารถอ้างถึงไฟล์ CONSTANT.TXT และคู่มือของ Visual Basic สำหรับสารสนเทศเพิ่มเติม

การทำให้เกิด Help

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

แอปพลิเคชัน Help ของ Windows ตามที่แสดงในรูปที่ 2.35 เกิดขึ้นเมื่อกำหนดค่า 6 ให้แก่คุณสมบัติ Action เราสามารถตั้งคุณสมบัติต่อไปเพื่อควบคุมลักษณะของไดอะล็อกบ็อกซ์ที่ปรากฏออกมา



รูปที่ 2.35
แอปพลิเคชัน Help

คุณสมบัติ

HelpCommand คุณสมบัตินี้กำหนดชนิดของความช่วยเหลือที่ต้องการให้เกิดขึ้น ตามปกติเราจะใช้ค่า

คงตัว **HELP__KEY** ซึ่งแสดงความช่วยเหลือสำหรับค่าหลักเฉพาะการนิยาม

HELP__KEY ได้ลงไว้ใน **CONSTANT.KEY**

HelpFile คุณสมบัตินี้กำหนดชื่อไฟล์ของ Help

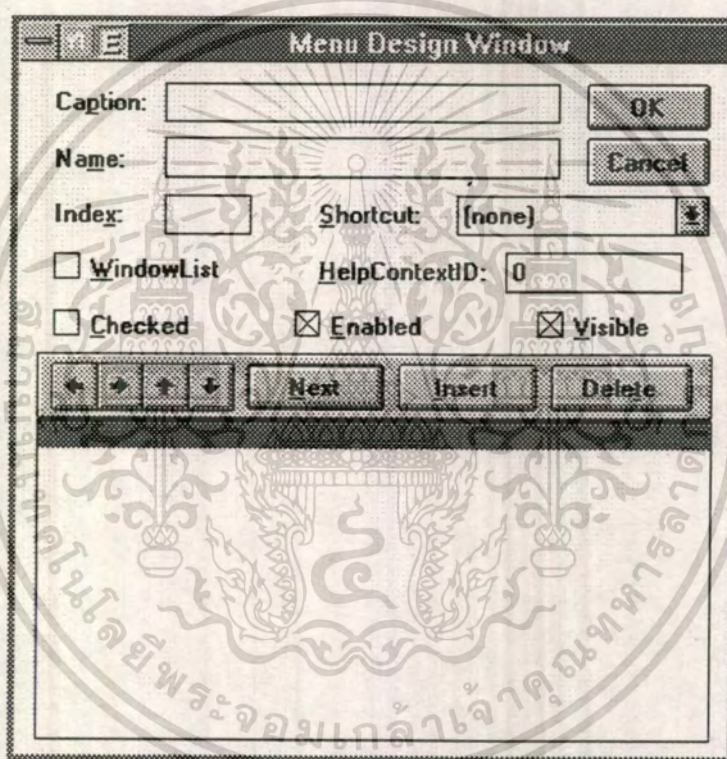
HelpKey คุณสมบัตินี้กำหนดค่าหลักที่ควรจะไปใช้กับจอภาพ Help ในตอนแรกเริ่ม

เมนู

แม้ว่าในตอนที่คุณได้สร้างแอปพลิเคชันขนาดเล็กหลาย ๆ โปรแกรมขึ้น แต่คุณก็อาจรู้สึกว่ามีโปรแกรมใดที่จะดูเหมือน “แอปพลิเคชันจริง ๆ” เหตุใดจึงเป็นเช่นนี้ สาเหตุก็มาจากการที่แอป

พลีเคชันที่สร้างขึ้นนั้นไม่มีเมนูบาร์ อย่างไรก็ตาม คุณสามารถแก้ไขสถานการณ์นี้ได้เพราะ Visual Basic ให้แนวทางที่ใช้สร้างเมนูต่าง ๆ ได้โดยง่ายด้วย เมนูดำเนินการด้วยแนวทางแบบอาศัยอีเวนต์ คล้ายกับออบเจกต์ที่เราพิจารณามาแล้ว

ในการที่จะสร้างเมนูสำหรับแอปพลิเคชันหนึ่ง ๆ คุณจะต้องใช้วินโดว์ Menu Design ของ Visual Basic ตามที่แสดงอยู่ในรูปที่ 2.36 สร้างโครงการใหม่ขึ้นหนึ่งโครงการ และเปิดวินโดว์นี้โดยการเลือกคำสั่ง Menu Design จากเมนู Window หรือโดยการกดปุ่มเมาส์บนปุ่ม Menu Design Window บนทูลบาร์ของ Visual Basic



รูปที่ 2.36

วินโดว์ออกแบบเมนูของ Visual Basic

คุณสร้างเมนูบาร์ของแอปพลิเคชันหนึ่ง ๆ ด้วยการสร้างลำดับชั้นของชั้นข้อมูลในวินโดว์ Menu Design สำหรับชั้นเมนูแต่ละชั้น คุณจะต้องนิยามคุณสมบัติ Caption และคุณสมบัติ Name ข้อความจำเป็นข้อความที่จะปรากฏขึ้นในเมนูบาร์หรือบนเมนูแบบดิ่งลง ซึ่งของชั้นเมนูจะระบุชั้น

เมื่อนั้นไว้ในรหัสโปรแกรมของคุณ การที่จำเป็นต้องใช้ชื่อก็เพราะว่าขึ้นเมนูแต่ละชั้นจะมีกระบวนการ Click

สมมติว่าคอมพิวเตอร์ของคุณประกอบไปด้วยฮาร์ดแวร์ที่จะแสดงภาพลักษณะของโทรทัศน์ภายในวินโดว์หนึ่ง ๆ เมนูสำหรับแอปพลิเคชันที่ควบคุมฮาร์ดแวร์อาจคล้ายกับเครื่องควบคุมระยะไกลของโทรทัศน์ เค้าโครงที่อาจเป็นไปได้สำหรับโครงสร้างเมนูที่มีสามระดับในลำดับชั้นเป็นดังต่อไปนี้

File

On

Off

Exit

Channel

Select..

Up

Down

Sound

Mute

Preset

Soft

Moderate

Loud

Louder

Softer

จุดไข่ปลา (...) ที่ปรากฏอยู่หลังขึ้นเมนู Select มีไว้เพื่อแสดงให้เห็นว่าการเลือกขึ้นเมนูนี้จะสร้างไดอะล็อกบ็อกซ์ขึ้น

เราลองมาทำให้เกิดส่วนเมนูของแอปพลิเคชันนี้ แต่ก่อนที่คุณจะย้ายไปยังวินโดว์ Menu Design คุณจำเป็นต้องจัดเก็ตรายการของเมนูเพิ่มเติมอีกเล็กน้อย

ในแอปพลิเคชันส่วนใหญ่ที่ดำเนินงานบน Windows คุณสามารถใช้คีย์บอร์ดเลือกขึ้นเมนูจากเมนูบาร์โดยไม่ต้องอาศัยเมาส์ เป็นลัดของคีย์บอร์ดนี้เรียกว่า แป้นเข้าถึง (access key) และ มักจะอาศัยการกดแป้น Alt และพิมพ์อักขระที่มีเส้นใต้ขีดบนเมนู เช่น ใน Visual Basic คุณสามารถพิมพ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Alt, F, S เพื่อเลือกคำสั่ง Save ของเมนู File ในวินโดว์ Menu Design คุณสามารถระบุเป็นเข้าถึงเพื่อใช้กับชั้นเมนูหนึ่ง ๆ ได้โดยการวางแอมเพอร์แซนด์ (&) ไว้ข้างหน้าอักขระที่ต้องการให้มีเส้นใต้ขีดอยู่บนเมนูของ

แอปพลิเคชัน (เครื่องหมายแอมเพอร์แซนด์ ไม่แสดงออกมาบนเมนู)

ให้พยายามให้อักขระตัวแรกของข้อความนำหน้าของชั้นเมนูเป็นเข้าถึง เว้นเสียแต่จะทำให้เกิดความสับสนขึ้น เช่น ในแอปพลิเคชันตัวอย่างนั้นทั้ง On และ Off ขึ้นต้นด้วยอักษรตัว o ฉะนั้นคุณจึงต้องเลือกอักขระตัวอื่นสำหรับชั้นเมนูตัวใดตัวหนึ่ง (ข้อยกเว้นประการหนึ่งเกี่ยวข้องกับการใช้ x ใน Exit แทนที่จะใช้ E การใช้ลำดับคีย์บอร์ด Alt, F, X เพื่อออกจากแอปพลิเคชันได้เริ่มกลายมาเป็นมาตรฐาน)

ต่อไปนี้เป็นหน่วยเมนูเพื่อใช้กับแอปพลิเคชันตัวอย่าง พร้อมกับได้ระบุอักขระที่เป็นแป้นเข้าถึงไว้ด้วย

&File

&On

Of&f

E&xit

&Channel

&Select...

&Up

&Down

&Sound

&Mute

&Preset

&Soft

&Moderate

&Loud

&Louder

&Softer

นอกเหนือจากข้อความจำหน้าและแป้นเข้าถึงแล้ว ขึ้นเมนูแต่ละชั้นยังต้องการชื่อแม้ว่าจะเป็นการมีเหตุผลที่จะใช้คำเดียวกันเป็นทั้งชื่อและข้อความจำหน้า แต่ก็มีบ่อยครั้งที่ไม่สามารถทำเช่นนั้นได้ เช่น *Exit* เป็นคำหลักของ Visual Basic และสามารถนำมาใช้เป็นข้อความจำหน้า แต่ไม่สามารถใช้เป็นของออบเจกต์ในรหัส

ในการตั้งชื่อนี้เป็นการดีที่สุดที่จะใช้แบบแผนมาตรฐานที่จะทำให้คุณสามารถรับรู้ชั้นเมนูแต่ละชั้นได้โดยทันที การใช้ชื่อที่มีความชัดเจนเป็นสิ่งสำคัญเพราะชั้นเมนูแต่ละชั้นมีกระบวนการงาน *Click* เกี่ยวข้องอยู่ด้วย ซึ่งในเวลาที่คุณกำลังทำงานรหัสจะเป็นการง่ายที่จะลืมว่ากำลังทำงานอยู่กับชั้นเมนูใด ในกรณีของโครงการตัวอย่างที่เสนอในที่นี้ ลองพยายามใช้คำเสริมหน้า *mnu* สำหรับชื่อออบเจกต์แต่ละชื่อ การใช้คำเสริมหน้านี้ก็เพื่อระบุออบเจกต์ว่าเป็นเมนู ซึ่งต่างไปจากปุ่มคำสั่งหรือสกรอลล์บาร์

สำหรับชั้นเมนูที่อยู่ในระดับแรกของลำดับชั้น ให้ตามหลังคำเสริมหน้า *mnu* ด้วยข้อความจำหน้า เช่น *mnuFile* สำหรับชั้นเมนูในระดับที่สองให้ใช้ *mnu* แล้วตามด้วยคำย่อที่มีอักษรสามตัวที่แสดงชั้นงาน เมนูระดับแรกที่สูงขึ้นเมนูระดับที่สองอยู่ในอันดับรอง แล้วตามด้วยข้อความจำหน้าของชั้นเมนูระดับที่สอง เช่น *mnuFilOn* หรือ *mnuChaSelect* ในทำนองเดียวกันนี้สำหรับชั้นเมนูในระดับที่สามก็ให้รวมเอาคำย่อสามตัวแรกของทั้งสองระดับแรกในกรณีของแอปพลิเคชันนี้ชื่อที่ใช้จะเป็นดังนี้

mnuFile
mnuFilOn
mnuFilOff
mnuFilExit
mnuChannel
mnuChaSelect
mnuChaUp
mnuChaDown
mnuSound
mnuSouMute
mnuSouPreset
mnuSouPreSoft
mnuSouPreModerate
mnuSouPreLoud

mnuSouLouder

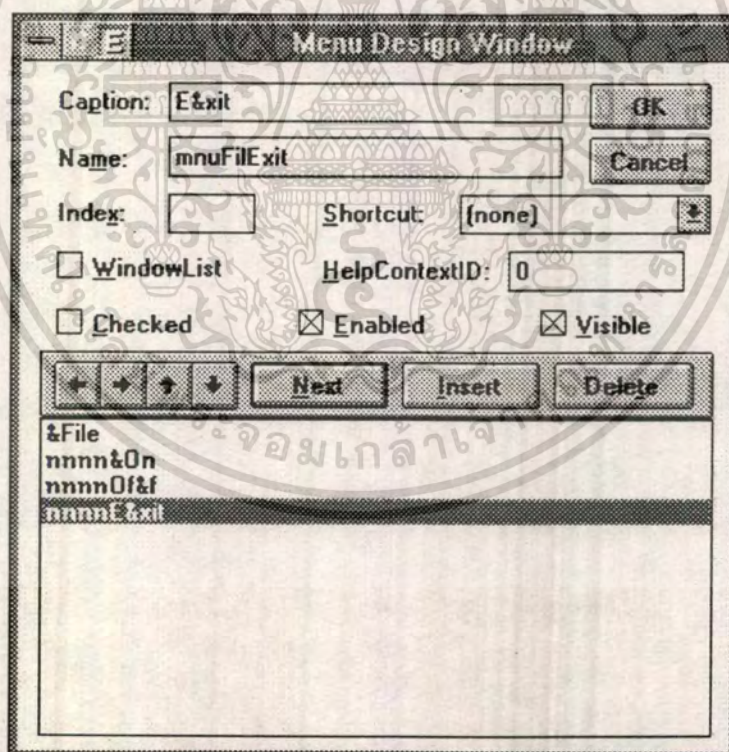
mnuSouSofter

ในตอนนี้คุณก็พร้อมที่จะย้ายไปสู่วินโดว์ Menu Design เพื่อสร้างเมนูสำหรับแอปพลิเคชันนี้

1. กดปุ่มเมาส์ในแท็บที่ชื่อของ Caption ของวินโดว์ Menu Design และพิมพ์ &File หลังจากนั้นกดเป็น Tab เพื่อเคลื่อนไปยังแท็บที่ชื่อ Name พิมพ์ mnuFile ในแท็บที่ชื่อนี้ และกดเป็น Enter ข้อความจำหน้า (แต่ไม่ใช่ชื่อ) จะปรากฏขึ้นในลิสต์ที่บริเวณส่วนล่างของวินโดว์

2. กดปุ่มเมาส์บนปุ่มลูกศรชี้ขวาที่อยู่เหนือลิสต์ที่ชื่อ จะปรากฏจุดสีจุดขึ้นในบรรทัดได้ &File ซึ่งแสดงว่าจะแทรกขึ้นเมนูถัดไปที่ระดับที่สองของลำดับชั้นเมนู

3. กลับไปยังแท็บที่ชื่อ Caption และพิมพ์ &On เคลื่อนไปยังแท็บที่ชื่อ Name พิมพ์ mnuFilOn และกดเป็น Enter ย้ายกลับไปยังแท็บที่ชื่อ Caption และพิมพ์ Of&f หลังจากนั้นเคลื่อนไปยังแท็บที่ชื่อ Name พิมพ์ mnuFilOff และกดเป็น Enter ทำกระบวนการอย่างเดียวกันนี้กับข้อความจำหน้า E&xit และชื่อ mnuFilExit วินโดว์ Menu Design ควรจะเป็นเช่นดังนี้

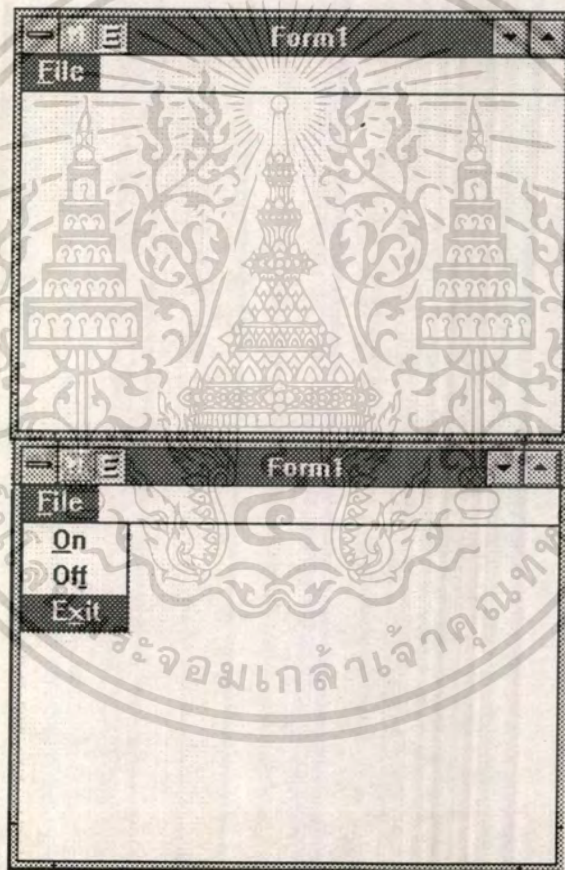


รูปที่ 2.37

แสดงหน้าต่างสร้างเมนู

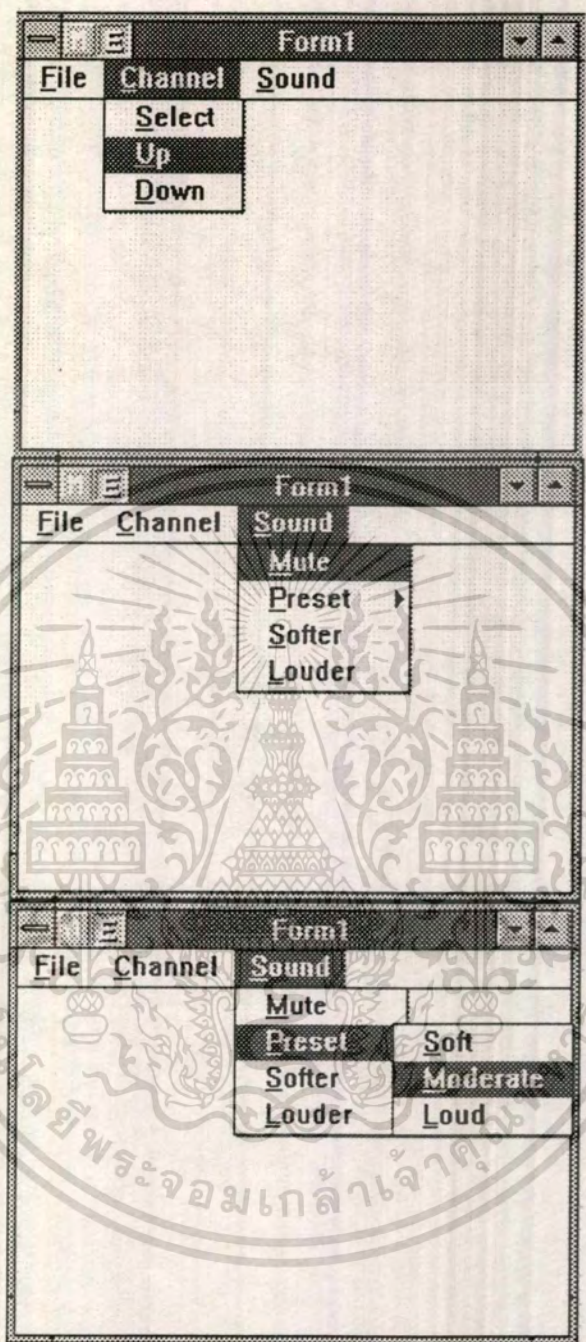
4. กดปุ่มเมาส์บนปุ่มลูกศรซ้ายเพื่อกลับไปยังระดับแรกของลำดับชั้น ในตอนนี้ป้อนส่วนที่เหลือของข้อความจำเป็นและชื่อจากรายการของคุณ ในการที่จะป้อนชั้นเมนูไปที่ระดับที่สามของลำดับชั้นเมนู ให้กดปุ่มเมาส์สองครั้งบนปุ่มลูกศรชี้ขวา ถ้าคุณวางชั้นเมนูในลำดับชั้นผิดพลาดไป คุณสามารถเลือกชื่อของชั้นเมนูในลิสต์ป๊อปอัพ และกดปุ่มเมาส์บนปุ่มลูกศรชี้ซ้ายและชี้ขวาเพื่อปรับระดับของชั้นเมื่อนั้น ปุ่มลูกศรขึ้นบนและชี้ล่างจะโยกย้ายชื่อไปยังตำแหน่งที่ต่างออกไปในรายการ

5. เมื่อคุณป้อนชั้นเมนูอย่างโดยสมบูรณ์แล้ว กดปุ่มเมาส์บนปุ่ม OK ในตอนนี้เมนูบาร์จะปรากฏอยู่บนแบบฟอร์มหลักของแอปพลิเคชัน คุณสามารถตรวจสอบชั้นเมนูด้วยการกดปุ่มเมาส์บนชื่อบนเมนูบาร์ เมื่อนั้นจะหย่อนลง แสดงหน่วยข้อมูลที่บรรจุอยู่ออกมาตามที่แสดงในรูปที่ 2.38



รูปที่ 2.38

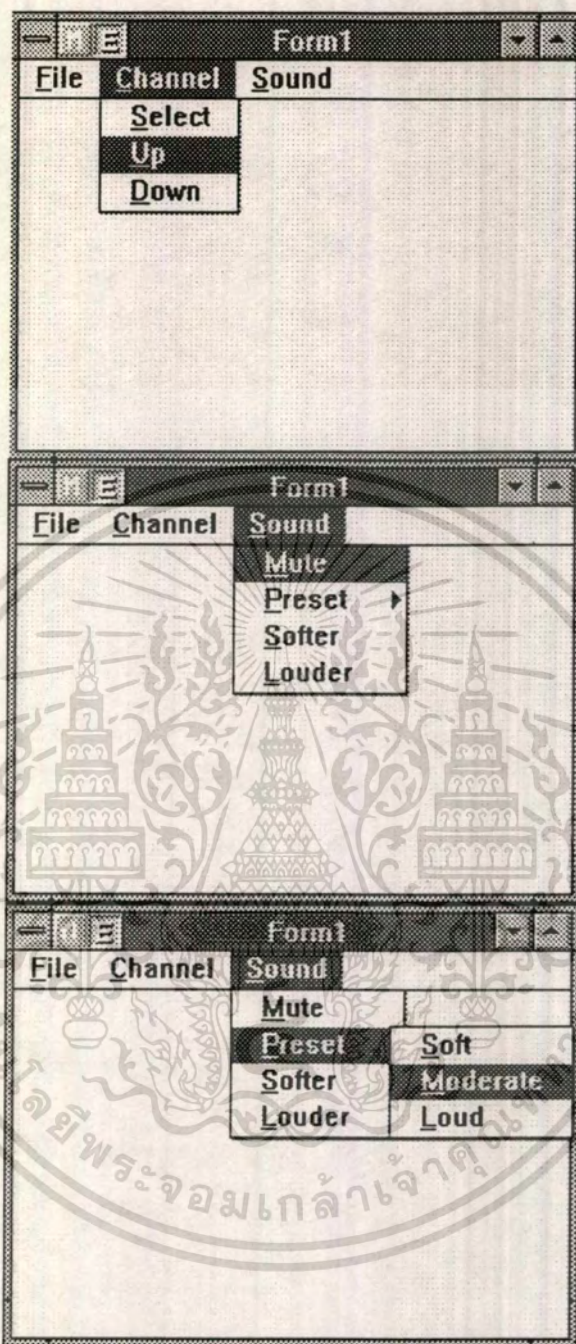
แบบฟอร์มกับเมนูบาร์และเมนูแบบดึงลง



รูปที่ 2.38

(ต่อ) แบบฟอร์มพร้อมกับเมนูบาร์และเมนูแบบคิ่งลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้



รูปที่ 2.38

(ต่อ) แบบฟอร์มพร้อมกับเมนูบาร์และเมนูแบบดึงลง

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ถ้าคุณกดปุ่มเมาส์บนหน่วยเมนูในแบบฟอร์ม Visual Basic จะเปิดวินโดว์รหัสสำหรับกระบวนการงาน Click ที่สอดคล้องกัน ในวินโดว์นี้คุณเขียนรหัสสำหรับการดำเนินการที่จะเกิดขึ้นเมื่อผู้ใช้เลือกชั้นเมนู เช่น กดปุ่มเมาส์บนชั้นเมนู Exit ของเมนู File เพื่อเปิดวินโดว์รหัสสำหรับกระบวนการงานอีเวนต์ `mnuFilExit_Click` คำสั่งของ Visual Basic ที่ใช้ยุติแอปพลิเคชันคือ End ดังนั้นกระบวนการงานสำหรับชั้นเมนู Exit ควรจะเป็นดังนี้

```
Sub mnuFilExit_Click()
```

```
End
```

```
End Sub
```

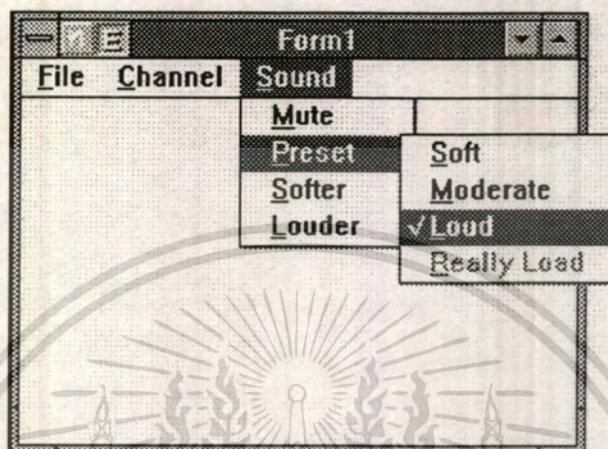
บ็อกซ์ใส่ข้อความสั่ง `Debug.Print` ในกระบวนการงานเมนูต่าง ๆ และหลังจากนี้ดำเนินการแอปพลิเคชันอีกครั้ง ในขณะที่เลือกชั้นเมนูต่าง ๆ นั้นให้จับตาที่วินโดว์ เพื่อเป็นการยืนยันว่ากำลังเฝ้าสังเกตกระบวนการงานเมื่อนั้นอยู่

ออกแบบเพิ่มเติมของวินโดว์ Menu Design

วินโดว์ Men Design ยังประกอบไปด้วยออกแบบที่น่าสนใจอีกจำนวนหนึ่ง เช่น ถ้าคุณป้อนไฮเฟน (-) เป็นข้อความจำหน้า Visual Basic ก็จะวางแท่งแยกเมนูเอาไว้ในเมนู ซึ่งเป็นการจัดกลุ่มชั้นเมนูที่สัมพันธ์กัน (คุณต้องกำหนดชื่อให้แก่น่วยข้อมูลไฮเฟน แม้ว่าจะไม่ได้นำไปใช้ในโปรแกรมของคุณก็ตาม) คุณยังสามารถกำหนดเป็นลัดให้แก่ชั้นเมนู เปิดวินโดว์ Menu Design อีกครั้งหนึ่งและเลือก E&xit หลังจากนั้นกดปุ่มเมาส์บนปุ่มเมาส์บนปุ่มลูกศรชี้ลงที่อยู่ด้านข้างบ็อกซ์ Shortcut คุณยังสามารถกำหนดการเคาะแป้นใด ๆ ที่ลงไว้ในเมนูแบบดั่งลงให้แก่ชั้นเมนู Exit

เช็บบ็อกซ์ Checked, Enabled และ Visible ในวินโดว์ Menu Design จะตรงกันกับคุณสมบัติของชั้นเมนูที่เลือกไว้ แต่ละคุณสมบัติสามารถตั้งให้เป็น True ตามที่แสดงด้วย X ในเช็บบ็อกซ์หรือ False (เช็บบ็อกซ์ว่าง ๆ) การตั้งคุณสมบัติ Checked ให้เป็น True เป็นการวางเครื่องหมายถูกเอาไว้บนเมนูข้างชั้นเมนู ซึ่งแสดงว่าออกแบบอยู่ในสภาพพร้อมใช้งาน (คุณสามารถเห็นเครื่องหมายถูกเช่นนั้นอยู่ถัดจากชั้น Tool Bar บนเมนู View ของ Visual Basic) คุณสมบัติ Enabled จะถูกตั้งให้เป็น True อย่างโดยปริยาย ถ้าคุณตั้งคุณสมบัตินี้เป็น False จะวาดชั้นเมนูเป็นสีเทา และไม่สามารถเลือกจากเมนู ถ้าคุณตั้งคุณสมบัติ Visible เป็น False ชั้นเมนูก็จะไม่ปรากฏบนเมนูเลย

รูปที่ 2.39 แสดงให้เห็นว่าเมนู Sound ของแอปพลิเคชันมีลักษณะเช่นใด ถ้าคุณเพิ่มเติมแท่งแยกสองแท่งไปที่ระดับแรก ตั้งคุณสมบัติ Checked ของชั้นเมนู mnuSouPreLoud ให้เป็น True เพิ่มเติมชั้นเมนูใหม่ชื่อว่า Really Loud และตั้งคุณสมบัติ Enabled ของชั้นเมนูใหม่ให้เป็น False



รูปที่ 2.39

เมนูพร้อมกับมีแท่งแยก เครื่องหมาย และอปชันสภาพใช้การไม่ได้

การตั้ง Checked, Enabled และ Visible ในวินโดว์ Menu Design แสดงถึงการตั้งคุณสมบัติที่เวลาเริ่มใช้งาน คุณสามารถเปลี่ยนแปลงการตั้งค่าต่าง ๆ เหล่านี้ในรหัสโปรแกรมของคุณด้วยความสั่งที่มีรูปแบบดังต่อไปนี้

`object.property = {True / False}`

อปชันอื่น ๆ ที่ปรากฏในวินโดว์ Menu Design เช่น Window List, Index และ HelpContextID เป็นคุณลักษณะระดับสูงที่อยู่นอกขอบเขตของหนังสือเล่มนี้ คุณสามารถพบสารสนเทศเกี่ยวกับคุณลักษณะเหล่านี้ได้ในคู่มืออ้างอิงของ Visual Basic

ป๊อปอัปเมนู

แม้ว่าตามปกติเมนูจะเกิดขึ้นจากเมนูบาร์ แต่ในบางครั้งอาจต้องการแสดงเมนูจากตัวควบคุมเมนูชนิดนี้เรียกว่า ป๊อปอัปเมนู (popup menu) ทั้งนี้เพราะตามปกติจะไม่เห็นแต่จะ “ผุดขึ้น” เมื่อกด

ปุ่มเมาส์บนตัวควบคุม เราสามารถจัดเตรียมเมนูใด ๆ ให้เกิดขึ้นในลักษณะผุดขึ้นโดยการใช้กลวิธี
PopupMenu

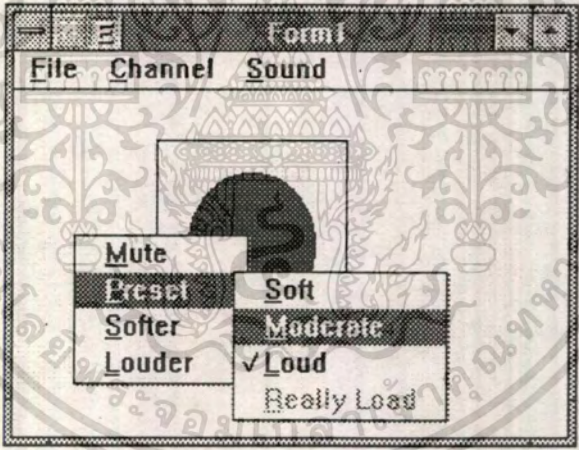
ตัวอย่างเช่น เราสามารถโยงเมนู Sound ตามที่แสดงในรูปที่ 5.45 เข้ากับสภาพของปุ่มควบคุม
โวลุ่ม ใช้เครื่องมือสร้างภาพเพื่อสร้างบ็อกซ์ภาพขึ้นบนจอภาพ และหลังจากนี้ใช้เครื่องมือสร้างรูปทรง
เพื่อวาดปุ่มควบคุมทรงกลมไว้ภายในบ็อกซ์ภาพ (ตอนแรกเริ่มรูปทรงจะเป็นสี่เหลี่ยมในการสร้างวง
กลมตั้งคุณสมบัติ Shape ให้เป็น 3- Circle) ป้อนรหัสต่อไปนี้สำหรับอีเวนต์ MouseDown ในบ็อกซ์
ภาพ

Sub Picture1_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

PopupMenu mnuSound

End Sub

เมื่อใช้กดปุ่มเมาส์ภายในบริเวณของบ็อกซ์ภาพ จะทำให้เกิดเมนู Sound ขึ้นตามที่แสดงไว้ใน
รูปที่ 2.40



รูปที่ 2.40

ป๊อปอัพเมนู

ตัวอย่างนี้เมนู Sound มีทั้งแบบผุดขึ้น และผ่านเมนูบาร์ ถ้าต้องการให้เมนูเห็นได้เฉพาะแบบผุดขึ้น จะต้องตั้งคุณสมบัติ Visible ให้เป็น False ในวินโดว์ Menu Design คุณสมบัติของชั้นที่อยู่ระดับบนสุด จะใช้ได้เฉพาะกับชั้นในเมนูบาร์ และจะถูกمينโดยกลวิธี PopuMenu

โครงสร้างฐานข้อมูล

การที่จะออกแบบและเรียกใช้ฐานข้อมูลได้อย่างมีประสิทธิภาพ เราควรที่จะมีความเข้าใจในหลักการพื้นฐาน (Concept) และโครงสร้างจริง ๆ (Physical) ของฐานข้อมูลนั้นให้ดีเสียก่อน ว่ามีลักษณะอย่างไร มีคุณสมบัติและความสามารถพิเศษอย่างไรบ้าง คำว่า "ฐานข้อมูล" ในที่นี้จะหมายถึง ฐานข้อมูลแบบตารางสัมพันธ์ (Relation Database) ไม่ว่าจะเป็นของ Access, dBase, Oracle หรือจะเป็นของใครก็แล้วแต่ ทุกตัวจะมีหลักการเหมือนกัน สิ่งที่แตกต่างกันจะเป็นเรื่องของโครงสร้างและการประยุกต์ใช้เทคนิคต่าง ๆ เพื่อให้สามารถจะจัดการกับฐานข้อมูลต่าง ๆ ได้เหมาะสมหรือดีเพียงใดเท่านั้น ซึ่งหากมีความเข้าใจในหลักการดังกล่าวเป็นอย่างดีแล้วก็จะช่วยการพัฒนาโปรแกรมให้ทำงานได้อย่างมีประสิทธิภาพมากกว่า

ฐานข้อมูลแบบสัมพันธ์

ฐานข้อมูลแบบสัมพันธ์มีหลักการพื้นฐานมาจากคณิตศาสตร์ ในเรื่องทฤษฎีของเซต(Theoty of set) โดยที่การจัดการหรือแสดงข้อมูลให้ผู้ใช้เห็นจะเป็นในรูปของตาราง

EMPLOYEE_ID	FIRST_NAME	LAST_NAME	SEX	DEPT_CODE
100	Wnat		M	001
102	Mary		F	001
103	Kartha		M	003
104	Michael		M	003

รูปที่ 2.41

แสดงตัวอย่างตารางข้อมูลพนักงาน

ข้อมูลประเภทต่าง ๆ จะแสดงอยู่ในรูปของตาราง (Table) โดยในแต่ละตารางจะเป็นการจัดเก็บรวบรวมข้อมูลประเภทเดียวกันเข้าไว้ด้วยกัน โดยที่แต่ละแถว (Row) ที่ประกอบขึ้นเป็นตารางนั้นก็คือ ฟิลด์ (Field) หรือ (Column) ซึ่งเป็นหน่วยที่แสดงคุณสมบัติ (Attribute) ของข้อมูลในแต่ละแถว จากตัวอย่างข้างต้นเป็นการแสดงตารางข้อมูลของพนักงาน โดยข้อมูลที่เก็บในตารางนี้จะมี เลขประจำตัวพนักงาน (EMPLOYEE_ID) ชื่อ (FIRST_NAME) นามสกุล (LAST_NAME) และเพศ (SEX)

โดยในแต่ละแถวจะเป็นข้อมูลพนักงานแต่ละคน ว่ามีหมายเลขประจำตัว และ ชื่อ - สกุล เป็นอย่างไร เช่นในแถวที่สอง เป็นข้อมูลพนักงานที่ชื่อ Jim Joojee เป็นผู้ชาย และมีเลขหมายประจำตัวคือ 101 เป็นต้น จะเห็นได้ว่า การที่จัดเก็บและนำเสนอข้อมูลให้อยู่ในรูปของตารางนั้น ทำให้ง่ายต่อความเข้าใจ เพราะเป็นรูปแบบที่หลาย ๆ คนคุ้นเคยอยู่เสมอ

โดยปกติแล้ว ในแต่ละตารางนั้นมักจะมีบางคอลัมน์หรืออาจหลายคอลัมน์ประกอบกัน ที่จะสามารถบอกถึงความแตกต่างกันของข้อมูลในแต่ละแถวได้ คอลัมน์ดังกล่าวจะเรียกว่าเป็น Primary Key เช่น ในกรณีของตารางข้อมูลพนักงานคอลัมน์ EMPLOYEE_ID ที่แสดงหมายเลขประจำตัวของพนักงาน จะเป็น Primary Key เพราะว่าพนักงานแต่ละคนจะมีหมายเลขประจำตัวที่แตกต่างกัน สามารถจะใช้เลขนี้ในการระบุว่าเป็นข้อมูลของพนักงานคนไหน เป็นต้น

เมื่อพูดถึงฐานข้อมูลแบบสัมพันธ์ (Relation Database) แล้ว ส่วนใหญ่จะประกอบไปด้วยตารางของข้อมูลต่าง ๆ ที่มีความสัมพันธ์กันเข้ามาไว้ด้วยกัน ตัวอย่างเช่นในตารางข้อมูลพนักงานข้างบน พนักงานแต่ละคนจะมีรหัสของหน่วยงานที่สังกัด (DEPT_CODE) หากต้องการทราบรายละเอียดของหน่วยงานที่ Jim Joojee สังกัดอยู่ก็ค้นหาได้จากตารางหน่วยงาน (รูป 2 - 1) เพื่อหาหน่วยงานที่มีรหัสเป็น 002 ตามข้อมูลรหัสหน่วยงาน (DEPT_CODE) ของ Jim ซึ่งจะเป็นหน่วยงานทางด้านการตลาด



รูปที่ 2.42

แสดงตัวอย่างตารางข้อมูลหน่วยงาน

หากเราทราบ DEPT_CODE หรือรหัสของหน่วยงานจากข้อมูลของพนักงานคนนั้น ก็สามารถใช้ข้อมูลนี้ในการค้นหารายละเอียดของหน่วยงานที่ต้องการได้ว่าเป็นหน่วยงานอะไร มีใครเป็นผู้จัดการ ดังนั้นในกรณีนี้ คอลัมน์ที่ชื่อ DEPT_CODE ของตารางข้อมูลพนักงาน จะเรียกว่าเป็นคอลัมน์ประเภท Foreign Key ซึ่งจะใช้เป็นการโยงความสัมพันธ์ไปยังคอลัมน์ DEPT_CODE ในตารางข้อมูลหน่วยงาน จะเป็น Primary Key ของตาราง และในทางกลับกัน ตารางข้อมูลหน่วยงานจะ

มีคอลัมน์ MGR_ID เป็น Foreign Key เพื่อโยงความสัมพันธ์ไปยังฟิลด์ EMPLOYEE_ID ที่เป็น Primary Key ของตารางข้อมูลพนักงานอีกที เพื่อสืบค้นหาว่าผู้จัดการของหน่วยงานนั้น ๆ คือใคร

นอกจากโครงสร้างพื้นฐานของฐานข้อมูลที่ใช้สำหรับการจัดเก็บข้อมูลแล้ว อีกองค์ประกอบหนึ่งที่สำคัญของระบบจัดการฐานข้อมูล ก็คือชุดคำสั่ง ที่ใช้ในการจัดการกับข้อมูลและโครงสร้างของฐานข้อมูล ซึ่งพอจะจัดกลุ่มชุดคำสั่งนี้ได้เป็น 2 ประเภทหลัก ๆ คือ

1.จัดการกับโครงสร้างของฐานข้อมูล เป็นการสร้าง ลบ หรือ แก้ไข ฐานข้อมูลและตารางเช่น การสร้างตารางข้อมูลพนักงานขึ้นมา โดยกำหนดว่าตารางต่อไปที่ต้องการจะประกอบไปด้วยคอลัมน์หรือฟิลด์อะไรบ้าง แต่ละคอลัมน์จะมีคุณสมบัติอย่างไร เช่น เป็นคอลัมน์ที่เก็บข้อมูลของตัวเลขหรือตัวอักษร เป็นต้น ชุดคำสั่งต่าง ๆ ที่เกี่ยวข้องกับการสร้างหรือเปลี่ยนแปลงโครงสร้างของฐานข้อมูล มีศัพท์เรียกกันว่า Data Definition Language หรือ DDL

2.จัดการกับข้อมูลที่อยู่ในตารางต่าง ๆ ในฐานข้อมูล เป็นการทำงานในเรื่องของ การเพิ่มเติม ลบ หรือ แก้ไข ข้อมูลในตารางต่าง ๆ เช่น การเพิ่มข้อมูลในพนักงานใหม่เข้าไปในตารางข้อมูลพนักงาน ลบออก หรือแก้ไขข้อมูลในบางคอลัมน์ของตารางตามที่ต้องการ เป็นต้น คำสั่งในประเภทนี้จะเรียกว่าเป็นคำสั่ง Data Management Language หรือ DML

ในปัจจุบันชุดคำสั่งสำหรับฐานข้อมูลเป็นที่นิยมกันมากเรียกว่า Structure Query Language (SQL) โดยจะประกอบไปด้วยคำสั่งสำหรับจัดการกับฐานข้อมูล ซึ่งจะมีมาตรฐานกำหนดไว้ ทั้งที่เป็นคำสั่งสำหรับจัดการกับโครงสร้าง (DDL) หรือจัดการกับข้อมูลในฐานข้อมูล (DML) ชุดคำสั่ง SQL ของผู้ผลิตแต่ละรายเกือบจะใกล้เคียงหรือเหมือน ๆ กัน แต่อาจจะแตกต่างกันบ้างเล็กน้อย เพราะผู้ผลิตแต่ละรายมักจะมีการเพิ่มเติมคำสั่งหรือตัวเลือกให้มากขึ้นกว่าที่กำหนดไว้ในมาตรฐาน เพื่อให้มีความสะดวกหรือมีประสิทธิภาพดีเหมาะกับฐานข้อมูลของตน

โครงสร้างของฐานข้อมูลใน JET 1.1

ระบบจัดการฐานข้อมูลของวิซวลเบสิกสำหรับในเวอร์ชัน 3 จะใช้โครงสร้างฐานข้อมูลแบบสัมพันธ์ ที่ได้นำเอามาจากชุดโปรแกรมจัดการฐานข้อมูล MS-Access หรือที่เรียกว่า JET 1.1 เพราะฉะนั้นฐานข้อมูลของ วิซวลเบสิกจึงมีความเหมือนกันและใช้ร่วมกันได้ (Compatible) กับฐานข้อมูลของ MS-Access 1.1 ทุกประการโดยมีโครงสร้างดังนี้

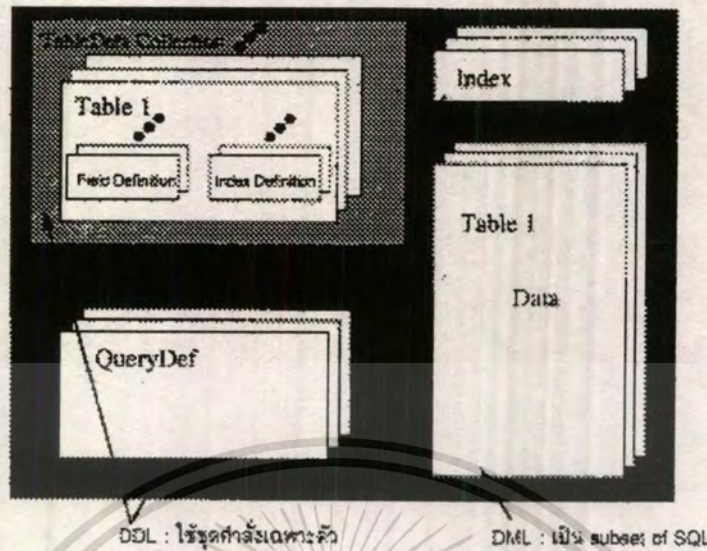
ฐานข้อมูล (Database) แต่ละตัวจะถูกจัดเก็บเป็นหนึ่งแฟ้มข้อมูล ซึ่งโดยปกติจะมีชนิดของแฟ้มเป็น .MDB (เราอาจกำหนดให้มีชนิดเป็นอย่างอื่นได้ตามต้องการและในหนึ่งแฟ้มหรือ

ฐานข้อมูลสามารถจัดเก็บฐานข้อมูลต่าง ๆ รวมทั้งดัชนี(Index) ได้มากกว่า 1 ตาราง ซึ่งมีลักษณะเช่นเดียวกับระบบจัดการฐานข้อมูลต่าง ๆ ทั่วไป แต่จะแตกต่างจากฐานข้อมูลของ dBase หรือ Foxpro ที่เราก่อนเคยซึ่งแต่ละตารางหรือดัชนีของ dBase หรือ Foxpro หนึ่งชุดนั้นจะเป็นหนึ่งแฟ้มข้อมูล

ดัชนี(Index) เป็นคุณสมบัติเพิ่มเติม ที่ระบบจัดการฐานข้อมูลทั่วไปต่างก็มีประยุกต์ใช้ เพื่อให้สามารถอ้างถึงตำแหน่งข้อมูลในตารางได้รวดเร็วขึ้น เพราะหากไม่มีดัชนีประกอบแล้ว การค้นหาข้อมูลในตารางจะต้องทำการอ่านข้อมูลในตารางทั้งหมดทีละแถว แล้วเปรียบเทียบกับเป็นแถวที่ต้องการหรือไม่ เช่นนี้ไปเรื่อย ๆ จนกว่าจะได้ข้อมูลตามที่ต้องการ ซึ่งจะเห็นได้ว่าเป็นกระบวนการที่ค่อนข้างจะเสียเวลามากทีเดียว และหากว่ามีการค้นหาข้อมูลที่ต้องการนั้นหลาย ๆ เทียว ก็จะต้องยิ่งเสียเวลาเพิ่มขึ้น

ถึงแม้ว่าระบบจัดการฐานข้อมูลจะมีการประยุกต์ใช้ดัชนีอย่างแพร่หลาย แต่ก็มีหลายรูปแบบที่แตกต่างกัน สำหรับ JET 1.1 ก็มีวิธีการประยุกต์ใช้ในรูปแบบของตัวเอง โดยดัชนีของ JET 1.1 จะเป็นลักษณะของการจัดเรียงลำดับของคอลัมน์ที่กำหนดให้เป็นดัชนี ซึ่งโดยมากจะกำหนดโดยยึดเอาวิธีในการเรียกหาข้อมูลที่ต้องการนั้น เช่น ค้นหาข้อมูลพนักงานโดยใช้ EMPLOYEE_ID เป็นค่าในการค้นหา ก็จะกำหนดดัชนีขึ้นมาโดยประกอบด้วยคอลัมน์ EMPLOYEE_ID เป็นต้น

ข้อมูลต่าง ๆ ที่จัดเก็บไว้ในฐานข้อมูล นอกเหนือจากข้อมูลของตารางต่าง ๆ เช่น ข้อมูลพนักงานแต่ละคน หรือข้อมูลรายละเอียดของหน่วยงานต่าง ๆ ที่เราพอมองเห็นเพราะเกี่ยวข้องกับการทำงานของเราแล้ว ยังจำเป็นต้องมีองค์ประกอบสำหรับการเก็บข้อมูลของโครงสร้างเพื่อจะได้บอกได้ว่า ฐานข้อมูลนี้ประกอบด้วยตารางอะไรบ้าง และในแต่ละตารางจะมีฟิลด์หรือตารางอะไรบ้าง ตารางนั้นมีดัชนีประกอบด้วยหรือไม่ เป็นต้น ดังนั้นโครงสร้างฐานข้อมูลแต่ละตัวของ JET 1.1 จะมีลักษณะและองค์ประกอบต่าง ๆ ดังแสดงในรูป 2.43



รูป 2.43

แสดงโครงสร้างและส่วนประกอบของฐานข้อมูลใน JET 1.1

องค์ประกอบต่าง ๆ ของโครงสร้างฐานข้อมูลใน JET 1.1

1. TableDefs Collection เป็นส่วนหนึ่งของการใช้นิยามองค์ประกอบของตารางต่าง ๆ ในฐานข้อมูลนั้น ซึ่งแต่ละตารางจะมีข้อมูลเกี่ยวกับโครงสร้างตารางนั้นอยู่แยกกันไปโดยองค์ประกอบของตารางจะมี 2 ส่วนคือ

1.1 Fields Definition คือส่วนที่เก็บรายละเอียดว่า ตารางนั้น ๆ ประกอบด้วยคอลัมน์อะไรบ้างแต่ละคอลัมน์มีลักษณะเป็นอย่างไร เช่น เป็นตัวเลข หรือ ตัวอักษร หรืออื่น ๆ เป็นต้น

1.2 Indexes Definition หากตารางนั้นมีดัชนี (Index) ประกอบเพื่อให้เข้าถึงตำแหน่งของข้อมูลที่ต้องการในตารางได้เร็วขึ้น ข้อมูลเกี่ยวกับดัชนีต่าง ๆ ของตารางจะเก็บไว้ในส่วนนี้ เช่น ดัชนีมีชื่อเรียกว่าอะไรบ้าง ประกอบด้วยคอลัมน์อะไรบ้าง และมีค่าซ้ำกันได้หรือเปล่า (Duplicated Key)

2. QueryDef จะเป็นส่วนที่เก็บคำสั่งสำหรับการจัดการข้อมูลในตารางต่าง ๆ ของฐานข้อมูลนี้ไว้สำหรับการเรียกใช้ในภายหลัง โดยจะเป็นคำสั่งใด ๆ ของ SQL ก็ได้ ไม่ว่าจะเป็นคำสั่ง Select, Insert, Delete หรือ Update ซึ่งโดยมากมักจะเก็บคำสั่งที่ซับซ้อนๆ เพื่อเป็นการแยกคำสั่งที่ใช้ในการจัดการกับข้อมูลออกจากตัวโปรแกรม โดยผูกคำสั่งในการจัดข้อมูลเข้ากับข้อมูลนั้น ซึ่งจะช่วยลดความยุ่งยากของโปรแกรมไปได้เป็นอย่างดี เพราะเมื่อต้องการใช้ก็เพียงแค่เรียกคำสั่งที่ต้องการขึ้นมาทำงานได้ทันที

ที่ ข้อสังเกตคือ QueryDef นี้จะไม่มีการเก็บรวมเป็น QueryDef Collection เหมือนอย่าง TableDef Collection แต่จะแยกเป็น QueryDef แต่ละตัวไปเลย

การจัดการกับโครงสร้างฐานข้อมูล

วิธีการจัดการกับโครงสร้าง Visual Basic มองแต่ละองค์ประกอบในโครงสร้างเป็น Object หนึ่ง ที่จะมีคุณสมบัติ (Property) และฟังก์ชันการทำงานเป็น Object เช่นเดียวกับ Object หรือ คอนโทรลอื่น ๆ ใน วิวลเบสิก การเรียนรู้ถึงลักษณะและความหมายของเจ็คต่าง ๆ ที่เกี่ยวข้องกับฐานข้อมูล จะช่วยให้เรามีความเข้าใจในองค์ประกอบต่าง ๆ และการจัดการฐานข้อมูลได้ดียิ่งขึ้น

การสร้างฐานข้อมูลเพื่อจะนำมาใช้ในวิวลเบสิกทำได้ 3 วิธีการหลัก ๆ ดังนี้คือ

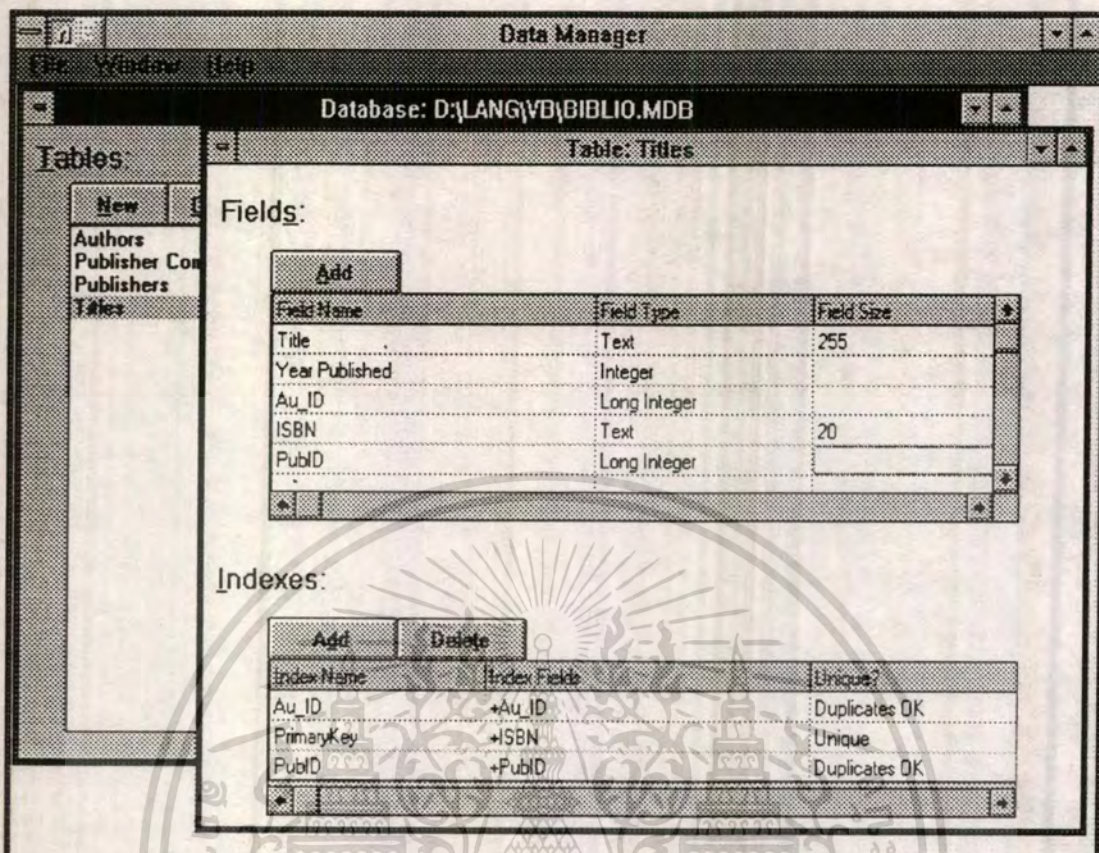
1. สร้างโดยอาศัยเครื่องมือของระบบจัดการฐานข้อมูลนั้น

วิธีนี้จะแตกต่างกันไปตามแต่รูปแบบของของฐานข้อมูลที่ใช้ เช่น หากต้องการใช้ฐานข้อมูลของ dBase ก็ใช้เครื่องมือของโปรแกรม dBase เป็นตัวสร้าง หรือหากต้องการเป็นฐานข้อมูลของ Paradox ก็ใช้โปรแกรมของ Paradox สร้างขึ้น และอื่น ๆ เป็นต้น

2. ใช้โปรแกรม Data Manager

โปรแกรม Data Manager จะมีมากับวิวลเบสิกแล้ว โปรแกรมนี้จะใช้สร้างฐานข้อมูลในรูปแบบของ JET 1.1 หรือ Access ได้เพียงอย่างเดียวเท่านั้น โดยที่เราไม่จำเป็นต้องมีโปรแกรม MS-Access ตัวโปรแกรม Data Manager นั้นอาจจะมีความสามารถจำกัด ทำได้เพียงการสร้าง ลบ และแสดงรายละเอียดของตารางและครรชนิต่าง ๆ ในฐานข้อมูลเท่านั้น ไม่สามารถทำการแก้ไขหรือเปลี่ยนแปลงรายละเอียดของตารางเดิมที่มีอยู่แล้วได้

โปรแกรม Data Manager นี้ เราสามารถเรียกขึ้นมาทำงานได้ระหว่างการใช้วิวลเบสิกพัฒนา งาน โดยเรียกจากคำสั่ง Data Manager ในเมนู Windows เมื่อเรียกเสร็จแล้วโปรแกรม Data Manager นี้จะทำงานในอีกวินโดว์หนึ่งที่เป็นอิสระจากโปรแกรมวิวลเบสิก ดังในรูปที่ 2.44



รูปที่ 2.44

แสดงวินโดวของโปรแกรม Data Manager

เมื่อต้องการสร้างฐานข้อมูลใหม่ DataManager จะสร้างได้เพียงแต่ฐานข้อมูลของ Access ซึ่งอาจจะเป็นเวอร์ชัน 1.0 หรือ 1.1 แต่หากเป็นการเปิดฐานข้อมูลเดิมเพื่อที่จะดูโครงสร้างหรือข้อมูลต่าง ๆ ที่มีอยู่ของฐานข้อมูล ตัวโปรแกรมนี้ก็จะเปิดฐานข้อมูลในรูปแบบอื่น ๆ ตามที่วิซวลเบสิกรู้จักได้

3.ใช้ Data Control

Data Control ก็นับเป็นคอนโทรลประเภทหนึ่งที่สามารถจัดวางลงในฟอร์มของวิซวลเบสิกได้ และใช้ในการจัดการกับข้อมูลต่าง ๆ ในฐานข้อมูลได้โดยไม่ต้องเขียนโปรแกรม วิธีนี้ถึงแม้จะสะดวกดีแต่ยังมีข้อจำกัดอยู่บ้าง

4.ใช้ชุดคำสั่งในการจัดการฐานข้อมูล(DDL)กับออบเจ็กต์ฐานข้อมูลต่าง ๆ

วิธีการนี้จะเหมาะสำหรับเมื่อเราต้องการจะสร้างฐานข้อมูลขึ้นมาใหม่ในระหว่างการทำงานของโปรแกรม จัดเป็นวิธีที่ยุ่งยากมากที่สุด แต่ให้ความยืดหยุ่นดี เหมาะสมที่จะสร้าง ลบ หรือ

แก้ไขรายละเอียดต่าง ๆ ได้ในระหว่างการทำงาน และยังใช้ในการจัดการกับออบเจ็กต์แบบที่เป็น QueryDef ได้ด้วยเช่นกัน

ชุดคำสั่งสำหรับการจัดการโครงสร้างฐานข้อมูล

จากพื้นฐานของวิซวลเบสิก ที่ได้มีการนำเอาหลักการของออบเจ็กต์ (Object) เข้ามาประยุกต์ใช้ บ้างบางส่วน แม้ในเรื่องที่เกี่ยวข้องกับฐานข้อมูลด้วยเช่นกัน ดังนั้นทุก ๆ องค์ประกอบของฐานข้อมูล ในวิซวลเบสิก ไม่ว่าจะเป็นตัวฐานข้อมูล (Database) ตาราง (Table) ฟیلด์หรือคอลัมน์ (Field) และ ครรชนี (Index) ตามที่ได้อธิบายไว้แล้วในเรื่องของโครงสร้างและองค์ประกอบของฐานข้อมูลใน วิซวลเบสิก จะมีลักษณะเป็นออบเจ็กต์หมด

ในทุก ๆ ออบเจ็กต์เกี่ยวกับฐานข้อมูลในวิซวลเบสิก จะประกอบด้วย 2 ส่วนสำคัญ คือคุณสมบัติและฟังก์ชันการทำงานกับออบเจ็กต์แต่จะไม่มีเหตุการณ์เช่นออบเจ็กต์หรือคอนโทรลอื่น ๆ ของ วิซวลเบสิกที่เราคุ้นเคยกัน เพราะว่าออบเจ็กต์ทางด้านฐานข้อมูลไม่ได้เป็นออบเจ็กต์ที่ทำการติดต่อกับผู้ใช้เพียงแต่การจัดการต่าง ๆ ที่เกี่ยวข้องกับฐานข้อมูลจะต้องกระทำผ่านออบเจ็กต์ต่าง ๆ เหล่านี้เท่านั้น

ดังนั้นชุดคำสั่งที่ใช้ในการจัดการกับโครงสร้างของฐานข้อมูล ก็คือการเรียกใช้ฟังก์ชันการทำงาน (Method) ที่เกี่ยวข้องกับออบเจ็กต์ของฐานข้อมูลที่ต้องการนั่นเอง ออบเจ็กต์ด้านฐานข้อมูล (Data Related Objects)

เพื่อให้เข้าใจมากขึ้น เรามาดูกันว่าในฐานข้อมูลหนึ่งที่จะใช้งานหรือสร้างขึ้นมา นั้นจะ ประกอบไปด้วยออบเจ็กต์ด้านฐานข้อมูลอะไรบ้าง ซึ่งจะอธิบายในรายละเอียดเรียงตามลำดับของโครงสร้างฐานข้อมูลได้ดังนี้

ฐานข้อมูล (Database Object)

คือออบเจ็กต์ที่หมายถึงตัวฐานข้อมูลเอง โดยจะมีคุณสมบัติ (Property) ที่เกี่ยวข้อง เช่น ฐานข้อมูลชื่ออะไร (Name) สามารถแก้ไขฐานข้อมูลได้หรือไม่ (Updatable) วิธีการเรียงลำดับข้อมูล เป็นอย่างไร (CollatingOrder)

ตาราง (Table Object)

เป็นออบเจ็กต์ที่อธิบายถึงตารางที่ต้องการ คุณสมบัติต่าง ๆ ก็เช่น ชื่อของตาราง (Name) อนุญาตให้แก้ไขข้อมูลในตารางได้หรือไม่ (Updatable) วันที่สร้างตารางนั้น (DataCreated) วันที่มีการแก้ไขล่าสุด (LastUpdated) จำนวนเรคอร์ดหรือข้อมูลที่มีอยู่ในตาราง (RecordCount) เป็นต้น

ไดนาเซ็ท (DynaSet Object)

ไดนาเซ็ทในที่นี้ของวิซวลเบสิก หากจะเปรียบเทียบกับคล้าย ๆ กับความหมายของคำว่า View ในฐานข้อมูลขนาดใหญ่เช่นใน Oracle ก็จะเป็นลักษณะของตารางที่กำหนดขึ้นซ้อนบนตารางอีกทีหนึ่งโดยมีเงื่อนไข เช่น เลือกเอาเฉพาะเรคอร์ดหรือแถวของตารางข้อมูลพนักงานที่มี DEPT_CODE เป็นหน่วยงานการตลาดเท่านั้นหรืออาจจะเป็นการประกอบกัน (Join) ของตารางมากกว่าหนึ่งตารางก็ได้ เพื่อความสะดวกในการเรียกใช้ข้อมูล โดยไม่ต้องไปหาในหลาย ๆ ตาราง ก็เป็นอีกตัวอย่างหนึ่ง ซึ่งวิธีการกำหนดเลือกข้อมูลหรือประกอบตารางที่เกี่ยวข้องเข้าด้วยกัน จะเลือกใช้คำสั่ง (Select) ของ SQL เป็นตัวกำหนด

สแน็ปช็อต (SnapShot Object)

สแน็ปช็อตจะมีลักษณะคล้าย ๆ กับไดนาเซ็ท แตกต่างกันเพียงหลัก ๆ ก็คือ ตัวสแน็ปช็อตนั้นจะอนุญาตให้อ่านได้เพียงอย่างเดียว ส่วนไดนาเซ็ทนั้นจะอนุญาตให้แก้ไขได้(ยกเว้นในบางกรณีเท่านั้น)นอกจากนี้ถ้าข้อมูลจริงในตารางมีการเปลี่ยนแปลงใด ๆ จะไม่มีผลต่อข้อมูลในสแน็ปช็อต ถ้าเป็นไดนาเซ็ทแล้วการเปลี่ยนแปลงนั้นจะมีผลตามและแสดงให้เห็นทันที

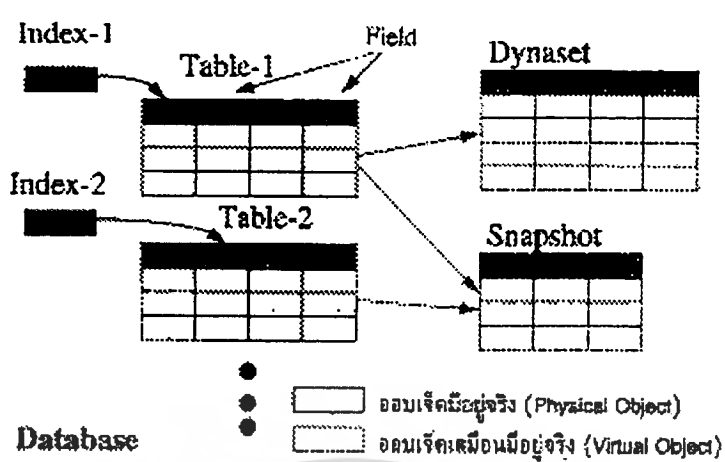
ฟิลด์ (Field Object)

ออบเจกต์ตัวนี้จะใช้ในเรื่องของโครงสร้างข้อมูลเท่านั้น คือใช้ในการเพิ่มฟิลด์หรือคอลัมน์เข้าไปในตาราง และการแก้ไขหรือลบออกเท่านั้น

ดรรชนี (Index Object)

ตัวออบเจกต์ดรรชนีก็เช่นเดียวกันกับออบเจกต์ฟิลด์ คือจะใช้ในเรื่องของโครงสร้างเท่านั้น จะไม่ได้ใช้ในการเรื่องของการเรียกใช้ข้อมูลในตารางเลย

ทั้งออบเจกต์และฟิลด์ดรรชนี ต่างก็เป็นส่วนประกอบย่อยของตาราง และตารางหลาย ๆ ตารางก็ประกอบรวมกันเป็นหนึ่งฐานข้อมูล



รูปที่ 2.45
แสดงส่วนประกอบต่าง ๆ ของฐานข้อมูล (Database) ชุดหนึ่ง

จากออบเจกต์ของฐานข้อมูลต่าง ๆ ที่อธิบายอย่างคร่าว ๆ ข้างบนนี้ ถ้าสังเกตสักหน่อยจะเห็นว่าทั้งออบเจกต์ตาราง ใดนาเซต และ สแน็ปช็อต จะมีลักษณะส่วนใหญ่คล้าย ๆ กัน คือต่างแสดงข้อมูลในรูปของตารางให้ผู้ใช้ได้เรียกใช้ ออบเจกต์ตารางคือตารางจริงของฐานข้อมูล ส่วนใดนาเซตและสแน็ปช็อตจะเหมือนกับตาราง (Virtual Table) คือจริง ๆ แล้วไม่มี แต่กำหนดขึ้นในขณะที่โปรแกรมกำลังทำงาน (run time) บนตารางอื่น ๆ อีกทีหนึ่ง ดังนั้นลักษณะวิธีการใช้งานของทั้ง 3 ออบเจกต์นี้เกือบจะคล้ายกันทั้งหมด ในคู่มือของวิซวลเบสิกหรือระบบความช่วยเหลือ (Help) บางครั้งจะเรียกรวม ๆ ว่าเป็น เรคอร์ดเซต (Record Set) ซึ่งในกรณีนี้จะหมายถึงออบเจกต์ใดใน 3 ออบเจกต์นี้ก็ได้อีก แต่หากเป็นเรื่องเฉพาะของออบเจกต์ใดออบเจกต์หนึ่ง ก็ระบุว่าเป็นออบเจกต์นั้น ๆ เลย

ออบเจกต์ต่าง ๆ ข้างบนที่ได้แนะนำมา เป็นเพียงออบเจกต์ที่เกี่ยวข้องหรือจำเป็นเมื่อเราต้องการจะทำงานกับข้อมูลที่อยู่ในฐานข้อมูล แต่ในที่นี้เราจะเน้นถึงเรื่องโครงสร้างของฐานข้อมูล ซึ่งมีออบเจกต์อื่น ๆ ที่เกี่ยวข้องเพิ่มเติมโดยออบเจกต์พวกนี้จะเรียกว่าเป็นประเภท (Collection) ของฐานข้อมูล โดยแต่ละ Collection จะเป็นเหมือนรายการที่แสดงว่าองค์ประกอบแต่ละประเภทมีอะไรบ้างในฐานข้อมูล เช่น ตาราง, ฟิลด์หรือคอลัมน์, ครรชนี เป็นต้นซึ่งจาก Collection ทั้งหมด เราก็สามารถจะบอกได้ว่า ในฐานข้อมูลนี้มีตารางอะไรบ้าง ในแต่ละตารางประกอบด้วยฟิลด์อะไร และมีครรชนีหรือไม่ ชื่ออะไร เป็นต้น คือจะเป็นส่วนที่เก็บรายละเอียดของโครงสร้างฐานข้อมูลทั้งหมด ออบเจกต์ประเภทนี้ที่เกี่ยวข้องมีดังนี้

TableDefs Collection

เป็นออบเจกต์ซึ่งเก็บข้อมูลที่แสดงว่าในฐานข้อมูลนั้น ๆ มีตารางอะไรบ้าง โดยในแต่ละรายการของ TableDefs Collection จะมีรายชื่อของตารางต่าง ๆ ที่อยู่ในฐานข้อมูล การเพิ่มหรือลบตารางในฐานข้อมูลก็ทำโดยการเพิ่มหรือลบชื่อตารางเข้าหรือออกจาก Table Collection ในฐานข้อมูลนั้น

Fields Collection

ในแต่ละตารางจะต้องประกอบไปด้วยฟิลด์หรือคอลัมน์ ดังนั้นในแต่ละตารางก็จะมี Field Collection ซึ่งแสดงรายการของฟิลด์ต่าง ๆ ที่มีอยู่ในตารางนั้น ๆ การเพิ่มหรือลบคอลัมน์เข้าหรือออกจากตาราง ก็ทำโดยการเพิ่มหรือลบข้อมูลใน Field Collection ของตารางนั้น ๆ

Indexes Collection

ในแต่ละตารางนอกเหนือจากจะมีฟิลด์แล้ว ยังอาจจะมีการกำหนดครรชนีหรือ Index ขึ้นเพื่อช่วยในการทำงานกับข้อมูลในแต่ละตารางให้รวดเร็วขึ้น ดังนั้นในแต่ละรายการของตารางจะมี Object Indexes Collection อยู่ด้วย ซึ่งในออบเจกต์นี้อาจจะมีรายชื่อครรชนีหรือไม่ก็ได้ ขึ้นอยู่กับว่าตารางนั้นมีครรชนีหรือไม่

การจัดการกับโครงสร้างของฐานข้อมูลจึงจะเกี่ยวข้องกับออบเจกต์ทั้งสามข้างต้นเป็นหลัก โดยมีขั้นตอนที่เกี่ยวข้องตามลำดับดังนี้

สร้างฐานข้อมูล

เราจะเริ่มต้นกันในขณะที่ยังไม่มีฐานข้อมูลที่ต้องการอยู่เลย และต้องการจะสร้างฐานข้อมูลนั้นขึ้นมา ซึ่งหลังจากที่เราได้ออกแบบของฐานข้อมูลที่ต้องการเรียบร้อยแล้ว ในขั้นตอนต่าง ๆ ของการสร้างฐานข้อมูลขึ้นมาใหม่จะมีดังนี้

1. ต้องกำหนดตัวแปรสำหรับฐานข้อมูลขึ้นมาก่อน โดยให้มีประเภทเป็นออบเจกต์ฐานข้อมูลหรือ Database ดังตัวอย่างที่กำหนดตัวแปร EmployeeDB

Dim EmployeeDB As Database

2. ใช้คำสั่ง Set ร่วมกับฟังก์ชัน CreateDatabase เพื่อกำหนดค่าให้กับตัวแปรชนิดฐานข้อมูลดังตัวอย่าง

Set EmployeeDB = CreateDatabase ("C:\SAMPLES\EMPLOYEE.MDB")

หากดูฟังก์ชันของ CreateDatabase ในรูปแบบเต็ม ๆ จะเป็นรูปดังนี้

Set database = CreateDatabase(databasename, locale [,options])

รายละเอียดของแต่ละส่วนที่ใช้ในการสร้างฐานข้อมูลของ CreateDatabase มีดังนี้
กำหนดชื่อแฟ้มของฐานข้อมูล (Databasename)

เป็นการระบุชื่อแฟ้มของฐานข้อมูล รวมทั้งตำแหน่งหรือไดเรกทอรีที่ต้องการ ในลักษณะเช่นเดียวกับการระบุชื่อแฟ้มข้อมูลทุกประการ เช่นตัวอย่างของเราข้างต้นจะสร้างฐานข้อมูลขึ้นมาอยู่ในแฟ้มที่ชื่อว่า Employee.mdb ซึ่งอยู่ในไดเรกทอรี \SAMPLES ของไดร์ C: เป็นต้น

กำหนดภาษาหรือรหัสของการเรียงลำดับ (locale)

ภาษาหรือรหัสของตัวอักษรที่ต่างกัน จะจัดเรียงลำดับของค่าต่าง ๆ ก็ต่างกันไปด้วย ส่วนนี้มิใช่ใช้เพื่อกำหนดภาษาหรือรหัสตามที่ต้องการ ซึ่งค่าที่เป็นไปได้มีหลายอย่างตามตารางต่อไปนี้

Constant Name	Value	Locate
DB_LANG_DUTCH	“;LANGID=0X0413;CP=1252;CO UNTRY=0”	Dutch
DB_LANG_GENERAL	“;LANGID=0X0809;CP=1252;CO UNTRY=0”	English, German, French
DB_LANG_ICELANDIC	“;LANGID=0X40F;CF;CP1252;CO UNTRY=0”	Icelandic
DB_LANG_NORWDAN	“;LANGID=0X0414;CP=1252;CO UNTRY=0”	Norwegian, Danish
DB_LANG_SPANISH	“;LANGID=0X040A;CP=1252;CO UNTRY=O”	Spanish, Italian
DB_LANG_SWEDFIN	“;LANGID=0X040C;CP=1252;CO UNTRY=O”	Swedish, Finnish

ค่าตัวเลือกของภาษาหรือรหัสที่จะกำหนดได้นั้น ไม่ได้มีสำหรับการจัดเรียงตามภาษาไทย ดังนั้นในการใช้งานสำหรับภาษาไทยของเราแล้ว ให้ละไว้ ไม่จำเป็นต้องกำหนดค่าในส่วนนี้ก็ได้ ซึ่งก็จะเป็นการใช้ภาษาตาม DB_LANG_GENERAL คือสำหรับภาษาอังกฤษทั่วไป

ตัวเลือกในการกำหนดลักษณะของฐานข้อมูล (option)

สำหรับในส่วนนี้จะกำหนดให้ผู้ใช้งานว่าต้องการสร้างฐานข้อมูลเข้ารหัสไว้หรือไม่รวมทั้งกำหนดว่าต้องการฐานข้อมูลตามรูปแบบของแอสเซสเวอร์ชัน 1.0 หรือ 1.1 ซึ่งกำหนดได้ตามตารางต่อไปนี้

Constant Name	Value	Option
DB_ENCRYPT	2	เข้ารหัสฐานข้อมูลใหม่ที่สร้าง
DB_VERSION10	1	สร้างฐานข้อมูลสำหรับ Access 1.0

เราสามารถจะเลือกค่าตัวเลือกดังกล่าวเป็นค่าใดก็ได้ หรือต้องการให้เป็นทั้งเวอร์ชัน 1.1 และเข้ารหัสด้วย จะกำหนดด้วยการบวกค่าทั้งสองเข้าด้วยกันเป็น DB_ENCRYPT + DB_VERSION10 ซึ่งหากไม่มีการกำหนดค่าของตัวเลือกนี้ ฐานข้อมูลที่สร้างได้จะเป็นแอสเซสเวอร์ชัน 1.1 และไม่มีการเข้ารหัส

3. ตรวจสอบผลของฟังก์ชัน CreateDatabase ว่าทำงานเรียบร้อยหรือไม่ ด้วยการเช็คค่าตัวแปรของฐานข้อมูลนั้นมีค่าหรือไม่

สร้างตารางใหม่

หลังจากที่ได้สร้างฐานข้อมูลมาแล้ว ฐานข้อมูลที่ได้ยังจะว่างเปล่าอยู่เราต้องกำหนดต่อไปว่า ฐานข้อมูลที่ต้องการนั้นประกอบไปด้วยตารางอะไรบ้าง ซึ่งจะต้องเข้าไปเกี่ยวข้องกับอบเจกประเภท Collection ของฐานข้อมูลทั้งสามตามทฤษฎีแล้วข้างต้นสำหรับขั้นตอนของการเพิ่มตารางใหม่เข้าไปในฐานข้อมูลมีดังนี้

1. สร้างตัวแปรสำหรับฟิลด์และกำหนดค่า

ตารางจะประกอบไปด้วยฟิลด์ต่าง ๆ ก่อนที่จะเริ่มสร้างตารางเราจะต้องสร้างหรือกำหนดฟิลด์ต่างๆ ที่ประกอบกันขึ้นเป็นตารางเสียก่อนดังนี้

- กำหนดตัวแปรประเภทตาราง โดยใช้คำสั่ง Dim พร้อมกับคำว่า New เพื่อเป็นการบอกให้สร้างตัวแปรนี้ขึ้นมาใหม่ (ใช้เฉพาะกับตัวแปรที่เป็นประเภทออบเจกเท่านั้น)

Dim EmployeesTB As New TableDef

- กำหนดตัวแปรประเภทฟิลด์ขึ้น โดยใช้คำสั่งเช่นเดียวกับการกำหนดตัวแปรสำหรับตารางและจะต้องกำหนดตัวแปรฟิลด์ 1 ตัว สำหรับฟิลด์หนึ่งทีประกอบขึ้นเป็นตารางดังตัวอย่าง

Dim EmpID As New Field

Dim FirstName As New Field

Dim LastName As New Field

- กำหนดค่าให้ตัวแปรประเภทฟิลด์ก่อน เพราะเป็นองค์ประกอบของตาราง ด้วยการเปลี่ยนค่าคุณสมบัติของตัวแปรฟิลด์

คุณสมบัติของฟิลด์ที่เกี่ยวข้องจะต้องกำหนดเช่น Name, Type, Size, และ Attribute ดังรายละเอียดต่อไปนี้

Name จะเป็นการกำหนดชื่อของฟิลด์สำหรับการอ้างถึง

Type เพื่อการกำหนดชนิดของข้อมูลที่จะเก็บในฟิลด์นี้ ว่าเป็นข้อมูลชนิดใดเป็นฟิลด์สำหรับเก็บตัวเลข ตัวอักษร และอื่น ๆ ซึ่งชนิดของข้อมูลจะกำหนดได้มีดังตารางต่อไปนี้

Size สำหรับกำหนดขนาดของฟิลด์ว่าจะมีความยาวเป็นเท่าใด โดยปกติมักจะไม่ต้องกำหนด เพราะชนิดของข้อมูลที่กำหนดในคุณสมบัติ ของ Type จะเป็นตัวบอกความยาวอยู่แล้ว ยกเว้นในกรณีของ DB_TEXT ที่ต้องระบุความยาวของฟิลด์นั้นด้วย

Attribute เป็นการกำหนดค่าคุณสมบัติเฉพาะสำหรับฟิลด์นั้น ๆ ว่าเป็นอย่างไร

วิธีกำหนดค่าให้กับคุณสมบัติ Attribute ของฟิลด์ นั้นทำได้เช่นเดียวกับการกำหนดค่าตัวเลือกอื่น ๆ คือสามารถจะกำหนดให้มีคุณสมบัติหลายประการรวมกันได้ โดยการนำค่าของตัวเลือกมาบวกกัน ตัวอย่างเช่น ถ้าจะให้ฟิลด์ที่ต้องการมี Attribute เป็นฟิลด์ที่มีความยาวคงที่และเพิ่มค่าของฟิลด์ไปอีกหนึ่งทุก ๆ ครั้งที่มีการเพิ่มเรคอร์ดใหม่ในตาราง ก็กำหนดดังนี้ DB_FIXEDFIELD + DB_AUTOINCRFIELD

* Emp_ID เป็นฟิลด์แบบ Counter คือเพิ่มค่าไปโดยอัตโนมัติเมื่อมีการเพิ่มเรคอร์ดใหม่

EmpID.Name = "Emp_ID"

EmpID.Type = DB_LONG

EmpID.Attributes = DB_AUTOINCRFIELD + DB_FIXEDFIELD

* First_Name เป็นฟิลด์ชนิดตัวอักษร มีความยาว 25 ไบท์

FirstName.Name = "First_Name"

FirstName.Type = DB_TEXT

FirstName.Size = 25

2. รวบรวมฟิลด์เข้าเป็นตาราง

หลังจากกำหนดค่าต่าง ๆ ของทุก ๆ ฟิลด์ที่ประกอบเข้าเป็นตารางเป็นที่เรียบร้อยแล้วให้เอาฟิลด์เหล่านั้นเพิ่มเข้าไปในตารางที่ต้องการตามลำดับดังนี้

กำหนดชื่อของตาราง (Table Name)

ด้วยการกำหนดในคุณสมบัติ Name ของตัวแปรรอบเจดตารางที่ได้กำหนดไว้

EmployeeTB.Name = "Employees"

ตัวอย่างข้างต้นเป็นการกำหนดชื่อของตารางที่ต้องการจะสร้างเป็น "Employees" ส่วนตัวแปร EmployeeTB เป็นตัวแปรรอบเจดตารางที่ได้กำหนดเอาไว้ก่อนหน้านี้แล้ว

เพิ่มฟิลด์ที่ต้องการเข้าไปในตาราง

การเพิ่มฟิลด์เข้าไปในตาราง ทำได้โดยการใช้ฟังก์ชันการทำงานกับออบเจกต์ (Method) Append เพื่อเป็นการเพิ่มรายการของฟิลด์เข้าไปในออบเจกต์ประกอบฟิลด์ ในที่นี้ก็คือ Fields ซึ่งหมายถึง Fields Collection ของตารางที่นำหน้าคือ EmployeeTB ดังตัวอย่าง

EmployeeTB.Fields.Append EmpID

EmployeeTB.Fields.Append FirstName

EmployeeTB.Fields.Append LastName

ถึงขั้นตอนนี้เราจะได้โครงสร้างของตารางที่ต้องการนั้นประกอบไปด้วยฟิลด์ต่าง ๆ โดยแต่ละฟิลด์มีคุณสมบัติตามที่กำหนด เช่น ชื่อ ประเภทของข้อมูล แต่ยังไม่ได้กำหนดว่า ตารางมีคีย์หลักอะไรบ้างหรือไม่

3. กำหนดดัชนี

หากต้องการจะให้ตารางมีดัชนีประกอบเพื่อให้การค้นหาข้อมูลทำได้เร็วขึ้น ต้องกำหนดดัชนีสำหรับตารางนั้น ๆ ด้วย ดังขั้นตอนต่อไปนี้

กำหนดตัวแปรประเภทออบเจกต์ดัชนี

กำหนดขึ้นโดยลักษณะเดียวกันกับการกำหนดตัวแปรสำหรับออบเจกต์ตารางหรือฟิลด์ใหม่ ดังตัวอย่าง

```
Dim Emprimary As New Index
Dim EmpNameIndex As New Index
```

กำหนดคุณสมบัติให้กับตัวแปรสำหรับออบเจกต์ดัชนี

คุณสมบัติที่ต้องกำหนดก็คือ Name, Fields, Primary, Unique ดังรายละเอียดต่อไปนี้

Name เป็นการกำหนดชื่อของดัชนีนั้น เพื่อใช้ในการอ้างถึงในภายหลัง ซึ่งทุกๆดัชนีจะต้องมีชื่อ

Fields กำหนดค่าในลักษณะของตัวอักษร ใช้สำหรับกำหนดว่าดัชนีที่ต้องการนี้ จะสร้างจากฟิลด์ใดในตารางบ้าง โดยจะต้องมีอย่างน้อยหนึ่งฟิลด์ แต่อาจจะมากกว่าก็ได้หากต้องการให้ประกอบด้วยหลายฟิลด์ให้ใช้เครื่องหมาย ' ; ' คั่นระหว่างฟิลด์ และสามารถจะกำหนดได้ว่าจะให้เรียงดัชนีบนแต่ละฟิลด์นั้นในลักษณะอย่างไร หากต้องการเรียงจากน้อยไปหามากให้ใส่เครื่องหมาย ' + ' นำหน้าฟิลด์นั้น แต่หากต้องการให้เรียงจากมากไปหาน้อยให้ใส่เครื่องหมาย ' - ' นำหน้าฟิลด์ที่ต้องการ

Primary กำหนดเพียงเป็นค่า True/False เท่านั้น เพื่อกำหนดว่าดัชนีที่ต้องการนี้ จะเป็นฟิลด์เดียวกับ Primary Key ของตารางหรือเปล่า เพราะถ้าเป็น Primary Key ค่าของฟิลด์ที่ประกอบกันเป็นดัชนีนี้ในแต่ละเรคอร์ดของตารางจะต้องไม่ซ้ำกันเลข และจะมีค่าเป็น Null ไม่ได้ และในแต่ละตารางจะมีดัชนี Primary Index ได้เพียงตัวเดียวเท่านั้น

Unique ให้กำหนดว่า True/False เท่านั้น เพื่อกำหนดว่าจะยอมให้ในดัชนีมีค่าซ้ำกันได้หรือไม่

4. เพิ่มตารางเข้าในฐานข้อมูล

ให้ใช้ฟังก์ชันการทำงาน (Method) เช่นเดียวกัน แต่ใช้กับตัวแปรออบเจ็กต์ฐานข้อมูลในส่วนขององค์ประกอบตาราง (TableDefs Collection) เพื่อเป็นการเพิ่มตารางเข้าไปในฐานข้อมูล โดย TableDefs Collection จะเป็นองค์ประกอบย่อยของฐานข้อมูล การอ้างถึงก็โดยเขียน TableDefs ต่อท้ายชื่อของตัวแปรออบเจ็กต์ฐานข้อมูลที่ต้องการ ดังตัวอย่าง

```
EmployeeDB.TableDefs.Append EmployeeTB
```

5. ปิดฐานข้อมูล

เช่นเดียวกับการทำงานกับแฟ้มข้อมูลโดยทั่วไป เมื่อเลิกหรือสิ้นสุดการทำงานกับแฟ้มข้อมูลนั้นแล้ว ควรจะต้องทำการปิดแฟ้มนั้นทุกครั้งเสมอไป วิธีการปิดฐานข้อมูลเมื่อเลิกใช้ ให้เรียกฟังก์ชันในการทำงาน (Method) Close บนออบเจ็กต์ฐานข้อมูลนั้น ดังตัวอย่าง

```
EmployeeDB.Close
```

ตัวอย่างต่าง ๆ ที่ยกมาแสดงข้างต้น แสดงการสร้างตารางเพียงตารางเดียวในฐานข้อมูลหากต้องการจะสร้างขึ้นหลาย ๆ ตารางก็กำหนดคำสั่งในขั้นตอน 2 ถึง 4 ข้างได้

การใช้วิซวลเบสิกกับฐานข้อมูลอื่น ๆ

เราสามารถวิซวลเบสิกในการจัดการฐานข้อมูลอื่น ๆ ได้ เช่น ฐานข้อมูลของ dBase III หรือ dBase IV ฐานข้อมูลของ FoxPro, Paradox หรือฐานข้อมูลภายนอกเช่น Oracle, SQLServer และอื่น ๆ ก็ได้ โดยคำสั่งหรือวิธีการจัดการกับฐานข้อมูลเป็นคำสั่งเดียวกัน แต่อาจจะแตกต่างกันที่การเปิดฐานข้อมูลที่ต้องมีการกำหนดค่าตัวเลือกเพิ่มเติมและอาจต้องมีการกำหนดค่าบางอย่างในแฟ้ม .ini ที่เกี่ยวข้องให้เหมาะสมด้วยเท่านั้น

การใช้วิซวลเบสิกกับ dBase และ FoxPro

ฐานข้อมูลของ dBase ที่จะใช้งานได้กับวิซวลเบสิก จะต้องเป็น dBase III,IV เท่านั้น ซึ่งวิธีการใช้ฐานข้อมูล dBase ก็มีจุดที่ต้องทำอยู่เพียง 2 - 3 จุดเท่านั้น ดังนี้

1. ต้องกำหนดค่าตัวเลือกใดเวอร์ที่ถูกต้อง

เมื่อต้องการจะใช้ฐานข้อมูลประเภทอื่น ๆ นอกเหนือจากเอกเซล เราจะต้องกำหนดในแฟ้ม VB.INI หรือ applicationname.ini ภายใต้ section ที่ชื่อ (Installable ISAMs) เพื่อเป็นการบอกว่าการจัดการกับข้อมูลประเภทนั้น ๆ จะต้องใช้ไครเวอร์ตัวไหน ดังนี้

จะเป็นแฟ้มข้อมูลแฟ้มหนึ่ง แต่เนื่องจากฐานข้อมูลในลักษณะที่วิซวลเบสิกรู้จักนั้น ประกอบไปด้วยหลาย ๆ ตารางและดัชนี ดังนั้นการนำเอาแฟ้ม .DBF และ .NDX หรือ .IDX มาไว้รวมกันภายใต้ไคลเรกทอรีหนึ่ง ก็เปรียบเหมือนกับการกำหนดให้ไคลเรกทอรีนั้นเป็นฐานข้อมูลอันหนึ่ง ที่ประกอบไปด้วยตารางและดัชนีต่าง ๆ อยู่ภายใต้ไคลเรกทอรีนั้น โดยชื่อของตารางและดัชนีที่เราอ้างถึงจะเป็นชื่อของแฟ้มข้อมูล สำหรับตารางหรือดัชนีนั้น ๆ

8. สร้างแฟ้ม tablename.INF เพื่อบอกความสัมพันธ์ว่ามีดัชนีอะไรประกอบอยู่บ้าง

เหตุที่ต้องกำหนดแฟ้มข้อมูล tablename.inf เพื่อบอกความสัมพันธ์ว่ามีดัชนีอะไรประกอบอยู่บนตารางนั้น ๆ บ้างซึ่งควรจะต้องมีการกำหนดในทุก ๆ ตารางที่มีดัชนี เพื่อให้วิซวลเบสิก และเรียกใช้ได้ และหากมีการเปลี่ยนแปลงหรือเพิ่มเติมข้อมูลในตารางใหม่ จะได้ปรับปรุงดัชนีที่กระทบให้ถูกต้องตามไปด้วย และแฟ้ม .INF นี้จะต้องอยู่ในไคลเรกทอรีเดียวกับแฟ้มข้อมูลอื่น ๆ วิธีการกำหนดแฟ้ม tablename.INF แสดงดังตัวอย่างต่อไปนี้

```
tablename.INF
NDX1=<index 1 filename>.NDX
NDX2=<index 2 filename>.NDX
.....
NDXn=<index n filename>.NDX
```

ตัวอย่างเช่นตาราง EMPLOYEE.DBF ประกอบด้วยดัชนี EMPID.NDX และ FNAME.NDX แฟ้ม .INF สำหรับตารางนี้จะเป็นดังนี้

```
EMPLOYEE.INF เป็นแฟ้มที่แสดงดัชนีประกอบบนตาราง EMPLOYEE
NDX1.EMPID.NDX
NDX2.FNAME.NDX
```

4. กำหนดส่วนของ databasename และ cocnnect ในคำสั่ง OpenDatabase ให้ถูกต้อง

นี่ในจุดที่ต้องแก้ไขในโปรแกรม ก็คือคำสั่งที่ใช้ในการเปิดฐานข้อมูลขึ้นมาใช้งาน ซึ่งเราจะต้องกำหนด 2 ส่วนคือ

```
SET database_object = OPENDATABASE (databasename, [,EXCLUSIVE [,
READONLY ,connect]]])
```

database_object ชื่อออเบเจ็กต์ฐานข้อมูล

databasename ให้ระบุเป็นชื่อไดเรกทอรีที่เก็บแฟ้มต่าง ๆ ประกอบกันเป็นฐานข้อมูล

connect เพื่อบอกว่าฐานข้อมูลที่ใช้เป็นฐานข้อมูลชนิดใด โดยกำหนดได้ดังนี้

'dBase III' เมื่อเป็นฐานข้อมูลของ dBase III

'dBase IV' เมื่อเป็นฐานข้อมูลของ dBase IV

'Foxpro 2.0' เมื่อเป็นฐานข้อมูลของ FoxPro 2.0

'Foxpro 2.5' เมื่อเป็นฐานข้อมูลของ FoxPro 2.5

ตัวอย่างเช่น หากต้องการจะใช้ฐานข้อมูลของ dBase III ที่เก็บรวมไว้ในไดเรกทอรี \PIS เราจะเขียนคำสั่ง Opendatabase ได้ดังนี้

```
Dim Pis As Database
```

```
Set Pis = Opendatabase (C:\PIS",,,,,,"dBaseIII")
```

การใช้วิซวลเบสิกกับฐานข้อมูล ODBC

มักจะใช้ในกรณีของการต่อกับระบบเน็ตเวิร์ก และฐานข้อมูลที่ต้องการจะใช้อยู่บนเครื่องที่เป็น SERVER ซึ่งอาจจะเป็นฐานข้อมูลของระบบใดก็ได้ ไม่ว่าจะเป็น SQL Server , Oracle หรืออื่น ๆ แต่ก่อนที่จะเรียกใช้ฐานข้อมูลนั้น ๆ บน SERVER จะต้องทำตามขั้นตอนต่าง ๆ ของการเรียกใช้ฐานข้อมูลภายนอกผ่านทาง ODBC ซึ่งมีรายละเอียดดังนี้

1. จะต้องมีการติดตั้งไดรเวอร์ ODBC สำหรับฐานข้อมูลภายนอกตัวนั้น ๆ ติดตั้งเสียก่อนจึงจะเรียกใช้ได้ สำหรับวิซวลเบสิกใน Version Professional Edition จะมีไดรเวอร์ ODBC สำหรับติดต่อกับฐานข้อมูลที่เป็น SQLServer หรือ Oracle ไว้ให้แล้ว แต่หากเป็นฐานข้อมูลของระบบอื่น ๆ จะต้องติดต่อขอไดรเวอร์ ODBC Setup ที่อยู่ในกรุปของวิซวลเบสิก ซึ่งหลังจากติดตั้งแล้วแฟ้ม ODBCINST.INI ในไดเรกทอรีของวินโดวส์จะเปลี่ยนไปดังนี้

```
[ODBC Driver]
```

SQL Server=Installed

Oracle=Installed

[ODBC Translators]

MS Code Page Translator=Installed

[SQL Server]

Drive=C:\WINDOWS\SYSTEMSQLSRVR.DLL

Setup=C:\WINDOWS\SYSTEMSQLSRVR.DLL

[Oracle]

Drive=C:\WINDOWS\SYSTEMSQORA.DLL

Setup=C:\WINDOWS\SYSTEMORASETUP.DLL

[MS Code Page Translator]

Translator=C:\WINDOWS\SYSTEMMSPXLT.DLL

Setup=C:\WINDOWS\SYSTEMMSPXLT.DLL

โดยในหัวข้อ (Section) ที่ชื่อ [ODBC Driver] จะแสดงรายการของไดรเวอร์สำหรับ ODBC ที่รู้จัก ซึ่งในตัวอย่างนี้จะมียู่ 2 รายการคือ SQL Server และ Oracle เท่านั้น ซึ่งชื่อของไดรเวอร์ ODBC ทั้ง 2 นี้จะถูกกำหนดขึ้นอีกทีในหัวข้อ (Section) แยกต่างหากสำหรับแต่ละตัวอีก เพื่อเป็นการบอกว่าเมื่อใช้งานข้อมูล ODBC ของระบบนั้น ๆ จะเรียกใช้ไดรเวอร์ที่จัดการจริง ๆ คือตัวใด ซึ่งในที่นี้จะเห็นว่ามี Section ที่ชื่อ [SQL Server] และ [Oracle] เพื่อบอกรายละเอียดอีกทีหนึ่ง

หากมี ODBC ไดรเวอร์อื่น ๆ ติดตั้งอีก เราจะเห็นชื่ออยู่ใน Section ของ [ODBC Driver] และมีหัวข้อ (Section) สำหรับไดรเวอร์นั้น ๆ อีกทีหนึ่งเช่นกัน

2. กำหนด Data Source สำหรับข้อมูลที่ต้องการจะทำงานด้วย เพื่อเป็นชื่อสำหรับการอ้างถึง และต้องกำหนดด้วยว่าเป็นฐานข้อมูล ซึ่งเราอาจจะกำหนดด้วยโปรแกรม Data Access Setup เช่นเดียวกับการติดตั้งไดรเวอร์ ODBC ซึ่งเมื่อกำหนดชื่อ Data Source นั้นอยู่ ดังแสดงในตัวอย่างต่อไปนี้

[ODBC Data Source]

Remote_PIS_DB=Oracle

.....

[Remote_PIS_DB]

Drive=C:\WINDOWS\SYSTEM\SQLORA.DLL

Description=Oracle on server

Server=R:server1

UserID=Rasanapa

หลังจากที่เรากำหนดชื่อ Data Source เรียบร้อยแล้ว เช่นในกรณีนี้ เราเพิ่มชื่อ Data Source เป็น Remote_PIS_DB และเป็นฐานข้อมูลของ Oracle ผลที่เกิดขึ้นกับแฟ้ม ODBC.INI ก็คือจะมีรายการชื่อ Remote_PIS_DB ในหัวข้อ (Section) ของ [ODBC DataSource] โดยจะมีค่าเป็น Oracle ซึ่งหมายถึงจะเป็นการอ้างถึงชื่อหัวข้อ [Oracle] ในแฟ้ม ODBCINST.INI เพื่อจะได้เรียกไดรเวอร์ขึ้นมาทำงานได้ถูกต้อง

นอกจากนั้นในแฟ้ม ODBC.INI ยังมีหัวข้อ (Section) ที่ชื่อ [Remote_PIS_DB] ซึ่งเป็นชื่อเดียวกันกับรายการที่แสดงใน [ODBC Data Source] ด้วย โดยจะบอกรายละเอียดของฐานข้อมูลนี้อยู่บน Server ชื่ออะไร เป็นต้น ข้อมูลในรายการของหัวข้อนี้อาจจะมีไม่เหมือนกันหากเป็นฐานข้อมูลคนละระบบ โดยในตอนที่เราติดตั้งและกำหนด Data Source โปรแกรมจะถามเองโดยอัตโนมัติ

3. ตั้งเปิดฐานข้อมูล (OpenDatabase) ซึ่งเราจะระบุชื่อของ Data Source ที่ต้องการใช้ แทนที่จะเป็นชื่อของแฟ้มของฐานข้อมูลดังปกติ และกำหนดค่าของ connect ด้วยดังนี้

Dim Remote_PIS As Database

```
Set Remote_PIS = OpenDatabase ("Remote_PIS_DB",,,, "ODBC;")
```

จากตัวอย่างจะเป็นการสั่งเปิดฐานข้อมูลจาก ODBC ที่ง่ายที่สุด โดยกำหนดชื่อ Data Source ตรงตำแหน่งที่เราใส่ชื่อเพิ่มของฐานข้อมูลในคำสั่ง OpenDatabase ซึ่งในกรณีนี้คือ Remote_PIS_DB และระบุว่าเป็น ODBC เท่านั้นที่ค่าของ connect

การสั่งแบบนี้เป็นการบอกให้ไปหาข้อมูลที่หัวข้อ [Remote_PIS_DB] ในแฟ้ม ODBC.INI หากข้อมูลยังไม่เพียงพอที่จะแสดงไดอะล็อกบ็อกซ์ได้ตอบ (Dialog Box) ขึ้นมาเพื่อให้ผู้ใช้ได้ตอบเพิ่ม ถ้าเป็นในกรณีนี้ก็จะแสดงไดอะล็อกบ็อกซ์เพื่อให้เติมรหัสผ่าน และหากเราไม่กำหนดชื่อของ Data Source ที่ต้องการ แต่ในส่วน connect ยังคงเป็น "ODBC;" เช่นเดิม ในกรณีนี้เมื่อเริ่มทำงาน โปรแกรมจะแสดงรายชื่อของ Data Source ที่รู้จักขึ้นมาให้เลือกว่าต้องการ Data Source ตัวใดกันแน่

ข้อมูลส่วนอื่น ๆ ที่ควรจะระบุลงไปในส่วนของ connect นั้นก็มี UID=username, PWD=password ซึ่งควรจะระบุไปด้วยเพื่อใช้ในการ LOGIN เข้าไปยังตัว Server หากไม่ต้องการจะคอยตอบรหัสผ่านทุกครั้งที่มีการเรียกใช้ และการใส่รหัสผ่านในโปรแกรมนี้จะเป็นการปลอดภัยกว่าการเก็บรหัสผ่านไว้ในแฟ้ม ODBC.INI ที่ใคร ๆ สามารถจะดูได้ สำหรับค่าอื่น ๆ ของส่วน connect ไม่ว่าจะเป็น DSN หรือ DATABASE เราควรจะกำหนดไว้ในแฟ้ม ODBC.INI จะดีกว่า เพราะจะทำให้ไม่สับสนในการเรียกใช้งาน

4. จากนั้นเราก็สามารถทำงานกับข้อมูลไม่จำเป็นการสอบถาม เพิ่ม ลบ หรือแก้ไขได้เช่นเดียวกับฐานข้อมูลของ JET 1.1 ทุกประการ

เพราะฉะนั้นโดยสรุปแล้ว ความสามารถของ ODBC ที่ใช้ติดต่อไปยังฐานข้อมูลในระบบอื่น ๆ จะช่วยให้โปรแกรมเราเป็นอิสระจากฐานข้อมูลได้มากขึ้น เราสามารถจะสร้างโปรแกรมขึ้นมาทดสอบกับฐานข้อมูลจำลองในเครื่องของเราเองก่อนได้ จากนั้นเมื่อนำไปใช้งานจริงกับฐานข้อมูลที่อยู่บน Server เราก็เพียงแต่เปลี่ยนที่คำสั่ง OpenDatabase เพียงคำสั่งเดียวเท่านั้น ก็เป็นอันเรียบร้อย แต่ที่ต้องระวังก็คือความถูกต้องและสัมพันธ์กันของแฟ้ม ODBCINST.INI และ ODBC.INI

การรับและการส่งออกผ่านไฟล์

ไฟล์เป็นเครื่องมือที่ใช้เก็บสารสนเทศระยะยาว ค่าที่บรรจุอยู่ในตัวแปรจะไม่คงทนถาวร โดยจะหายไปเมื่อหยุดดำเนินงานแอปพลิเคชัน และมีการคำนวณใหม่ในเวลาที่คุณดำเนินงานอีกครั้ง ในการที่จะเก็บข้อมูลให้คงทนโดยที่สามารถเข้าถึงและค้นคืนได้ จะต้องเก็บข้อมูลไว้ในไฟล์ ในบทนี้จะมาดูว่า Visual Basic จัดการกับการดำเนินการไฟล์มาตรฐานอย่างไร ซึ่งประกอบไปด้วย การกำหนดตำแหน่ง (locating) การเปิด การเก็บ และการปิดไฟล์ ในบทนี้จะสร้างแอปพลิเคชันง่าย ๆ ขึ้นหนึ่งโปรแกรมที่แสดงเครื่องมือระบบไฟล์ของ Visual Basic โดยอาศัยเครื่องมือเหล่านี้ จะทำให้สามารถสร้างไดอะล็อกบ็อกซ์ที่ทำงานคล้ายกับไดอะล็อกบ็อกซ์ File Open และ File Save ที่ใช้ในแอปพลิเคชันอื่น ๆ ที่เขียนเพื่อใช้งานกับ Windows นอกจากนี้เรายังมาดูแนวทางการเก็บและการเข้าถึงข้อมูลในไฟล์ นั่นคือกลวิธีแบบทีละหนึ่งบรรทัด (line by line method) ซึ่งจะปฏิบัติกับสารสนเทศในลักษณะเป็นข้อความ และกลวิธีแบบอิงเรคคอร์ด (record-oriented method) นอกจากนี้ยังจะได้เรียนรู้ว่าจะเก็บสารสนเทศโครงสร้างแบบในไฟล์ INI อย่างไรเพื่อที่ว่าจะสามารถเก็บสารสนเทศเริ่มใช้งานและโครงสร้างสำหรับแอปพลิเคชันด้วยแนวทางมาตรฐานของ Windows

ลักษณะทั่วไปของระบบไฟล์

ตามที่ได้อธิบายไว้แล้วว่าไมโครโปรเซสเซอร์ของคอมพิวเตอร์ใช้บิตของสารสนเทศที่เก็บอยู่ในหน่วยความจำเป็นคำสั่งและข้อมูล การออกแบบฮาร์ดแวร์ของไมโครโปรเซสเซอร์จะทำให้สามารถเข้าถึงแรม (หน่วยความจำเข้าถึงเชิงสุ่ม) ในระบบคอมพิวเตอร์ได้อย่างทันที แต่กระนั้น แม้แต่ไมโครคอมพิวเตอร์รุ่นใหม่ที่มีเมกะไบต์ของแรมนับโหลก็ไม่สามารถเก็บสารสนเทศทั้งหมดที่ต้องการไว้ในแรม ระบบต้องการตำแหน่งเก็บอื่น ๆ สำหรับข้อมูลที่ไม่สามารถบรรจุเข้าในแรม

หน่วยเก็บภายนอกที่ระบบคอมพิวเตอร์ใช้มีหลายรูปแบบเช่นบัตรเจาะรู ฮาร์ดดิสก์ ฟลอปปีดิสก์ และซีดีรอม สื่อภายนอกเหล่านี้สามารถเก็บข้อมูลได้นับร้อยเมกะไบต์ หรือแม้แต่เป็นกิกะไบต์ด้วยราคาซึ่งน้อยกว่าการเติมแรมนับร้อยเท่า ราคาและความจะทำให้หน่วยเก็บภายนอกเป็นที่น่าดึงดูดอย่างมาก แม้ว่าจะช้ากว่าและสะดวกน้อยกว่าก็ตาม

การที่หน่วยเก็บภายนอกสามารถจัดการกับข้อมูลได้ในปริมาณมาก ๆ ทำให้เกิดปัญหาเกี่ยวกับการจัดระบบขึ้น นักพัฒนาระบบคอมพิวเตอร์ในยุคก่อนแก้ปัญหาเกี่ยวกับการจัดระบบโดยอาศัยการเลียนแบบแนวทางของธุรกิจ โดยการสร้างสิ่งที่เทียบเท่ากับระบบไฟล์และเรียกกลุ่ม

ข้อมูลแต่ละกลุ่มที่แยกกันว่าเป็นไฟล์ ยูเซอร์อินเตอร์เฟซเชิงกราฟิกยังคงแสดงบทบาทการเลียนแบบนี้ โดยการใช้ไอคอนของไฟล์โฟลเดอร์และตู้เก็บไฟล์

เนื่องจากหน่วยเก็บดิสก์อยู่ภายนอกตัวประมวลผลหลัก ดังนั้นคอมพิวเตอร์จึงต้องทำกิจกรรมบางอย่างเพื่อรับสารสนเทศจากหรือเดิมสารสนเทศเข้าไปในไฟล์ Visual Basic จัดจำแนกไฟล์ตามแนวทางที่คอมพิวเตอร์จะสามารถเข้าถึงไฟล์ นั่นคือแบบลำดับ (sequential) แบบเข้าถึงเชิงสุ่ม (random) หรือแบบทวิภาค (binary)

1. ไฟล์ลำดับ เป็นไฟล์ข้อความแบบง่าย ๆ Visual Basic จะอ่านจากและเขียนไปยังไฟล์ลำดับทีละหนึ่งบรรทัด ด้วยเหตุนี้จึงได้ชื่อเช่นนี้ บรรทัดแต่ละบรรทัดอาจยาวหรือสั้นได้แล้วแต่ความต้องการ

2. ไฟล์เข้าถึงเชิงสุ่ม เป็นไฟล์ที่ Visual Basic สามารถอ่านและเขียนเนื้อหาเรียงตามลำดับตามที่ต้องการ แต่ละบรรทัด (หรือเรคคอร์ด) จะต้องมีความยาวเท่ากัน

3. ไฟล์ทวิภาค เป็นเพียงกลุ่มของไบนารี โดย Visual Basic จะไม่ตั้งข้อสันนิษฐานใด ๆ เกี่ยวกับแนวทางการจัดระบบข้อมูล ซึ่งทำให้โปรแกรมสามารถจัดระบบและเข้าถึงข้อมูลด้วยรูปแบบแล้วแต่ความต้องการ ไฟล์ชนิดนี้ให้ความยืดหยุ่นอย่างมาก แต่นั่นหมายถึงมีงานที่ต้องทำกับไฟล์มากขึ้น (การที่ต้องอาศัยความพยายามอย่างมากเช่นนี้ ในหนังสือจึงไม่ครอบคลุมไฟล์ทวิภาค)

การดำเนินการกับไฟล์

ในเวลาที่เปิดไฟล์หรือเก็บงานเข้าไว้ในไฟล์ โปรแกรมที่ดำเนินงานบน Windows ส่วนใหญ่จะนำเสนอไอคอนบ็อกซ์ที่สามารถระบุชื่อ ไดเรกทอรี และไครฟี่ของไฟล์ ใน Visual Basic คุณอาจใช้ Common Dialog Control เพื่อนำเสนอไอคอนบ็อกซ์ File Open หรือไอคอนบ็อกซ์ File Save As อย่างไรก็ตาม คุณอาจต้องการสร้างไอคอนบ็อกซ์ที่คล้ายคลึงกันแต่มีอุปชัณเพิ่มขึ้น เราสามารถสร้างไอคอนบ็อกซ์เช่นนั้นโดยอาศัยเครื่องมือสามอย่างที่อยู่ในชุดบ็อกซ์ของ Visual Basic นั่นคือเครื่องมือสร้างไฟล์ลิสต์บ็อกซ์ เครื่องมือสร้างไดเรกทอรีลิสต์บ็อกซ์ และเครื่องมือสร้างไครฟี่ลิสต์บ็อกซ์ ตามที่แสดงในรูปที่ 2.79 เครื่องมือเหล่านี้ทำให้สามารถสร้างออบเจกต์ได้สามชนิดที่ได้รับการออกแบบให้มีการโต้ตอบซึ่งกันและกัน แอปพลิเคชันส่วนใหญ่จะไม่ใช้ออบเจกต์เหล่านี้แยกกัน



รูปที่ 2.49

เครื่องมือสร้างไฟล์ลิสต์บ็อกซ์ เครื่องมือสร้างไดเรกทอรีลิสต์บ็อกซ์ และเครื่องมือไครฟี่ลิสต์บ็อกซ์

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ไฟล์ลำดับ

ตามที่เราได้กล่าวไว้ก่อนหน้านี้แล้วว่าวิธีการประมวลไฟล์ข้อมูลง่ายที่สุดที่เกี่ยวข้องกับการอ่านหรือการเขียนบรรทัดของข้อมูลที่มีลักษณะคล้ายกับบรรทัดของข้อความในหนังสือที่เรียงลำดับต่อเนื่องกัน เช่น แก้ไขข้อความดังเช่นแอปพลิเคชัน Notepad ของ Windows จะจัดการกับเอกสารด้วยรูปแบบที่เรียงลำดับต่อเนื่องกัน โปรแกรมจะอ่านบรรทัดแรกเข้าไปไว้ในหน่วยความจำ และแสดงบรรทัดนั้นออกมา หลังจากนั้นอ่านและแสดงบรรทัดที่สอง และเป็นเช่นนี้เรื่อยต่อเนื่องไป เมื่อบอกให้โปรแกรมเก็บเอกสารก็จะเกิดกระบวนการเดียวกันนี้ขึ้น ในขณะที่เขียนเอกสารเข้าไปไว้ในไฟล์ที่ละหนึ่งบรรทัด

สังเกตสัญลักษณ์ <CR> และ <LF> ที่อยู่ในรูปภาพ อักขระสองตัวนี้ซึ่งประดิษฐ์ขึ้นตั้งแต่มีการสื่อสารด้วยโทรพิมพ์ จะนำมาใช้แยกบรรทัดแต่ละบรรทัดในไฟล์ลำดับออกจากกัน โทรพิมพ์จะพิมพ์ลงบนม้วนกระดาษโดยจะพิมพ์ออกมาทีละหนึ่งบรรทัด นอกเหนือจากการพิมพ์ข้อความแล้วโทรพิมพ์ยังสามารถตอบสนองต่อรหัสควบคุมที่พิมพ์ไม่ได้บางตัว เช่น Ctrl-G ซึ่งจะสั่งกระดิ่งบนเครื่อง การเริ่มต้นข้อความบรรทัดใหม่ต้องอาศัยรหัสควบคุมสองตัวคือ อักขระปิดแคร์ (แอสกี 13) ซึ่งจะเคลื่อนย้ายหัวพิมพ์ไปที่จุดเริ่มต้นของบรรทัด และอักขระป้อนบรรทัด (แอสกี 10) ซึ่งจะเคลื่อนกระดาษขึ้นหนึ่งบรรทัดรหัสทั้งสองตัวนี้จะถูกบรรจุอยู่ในไฟล์ข้อความควบคู่กันกับข้อความ

เมื่อได้มีการพัฒนาอุปกรณ์ส่งออกใหม่ ๆ ขึ้น เช่น คัมปีเทอร์มินอลและเครื่องพิมพ์ไคชีวิล (Daisywheel) ผู้ผลิตอุปกรณ์เหล่านี้ก็ได้ยอมรับเอาการใช้อักขระควบคุมเหล่านี้มาด้วย โลกของ MS-DOS/Windows ก็ยังได้สืบทอดมรดกอักขระปิดแคร์และอักขระป้อนบรรทัดไว้ด้วย ดังนั้นเราจะพบอักขระคู่นี้บ่อยเป็น CR และ LF อยู่บ่อย ๆ

การเปิดและการปิดไฟล์ลำดับ

ข้อความสั่ง Open ใช้บ่งบอก Visual Basic ว่าจะเข้าถึงไฟล์ใด วากยสัมพันธ์ของข้อความสั่ง Open จะเป็นดังนี้

```
Open filename For {Input | Output} As # filenumber
```

นอกจากที่จะต้องให้ชื่อไฟล์แล้ว เราจะต้องบอก Visual Basic ด้วยว่าต้องการอ่านจากไฟล์ (การรับเข้าหรืออินพุต) หรือเขียนไปยังไฟล์ (การส่งออกหรือเอาต์พุต) และจะต้องกำหนด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

หมายเลขไฟล์ไว้ด้วย หมายเลขไฟล์จะต้องเป็นจำนวนเต็มในช่วง 1 ตลอดจนถึง 255 ต่อไปนี้เป็นตัวอย่างของข้อความสั่งที่เปิดไฟล์

Open "C:\CONFIG.SYS" For input As # 1

Doc\$ = "A\WATCH\BANZAI.DOC"

Open Doc\$ For input As # 2

Open "RESULT.TXT" For Output As # 5

เมื่อพยายามเปิดไฟล์สำหรับการรับเข้าโดยที่ไม่มีไฟล์นั้นปรากฏอยู่ Visual Basic จะก่อให้เกิดข้อความระบุมความผิดพลาดขึ้น ถ้ากำลังเปิดไฟล์สำหรับการส่งออก Visual Basic จะสร้างไฟล์ใหม่ขึ้นเสมอโดยจะเขียนทับไฟล์ใด ๆ ที่มีอยู่ที่มีชื่อเดียวกัน ถ้าต้องการตรวจสอบไฟล์ที่มีอยู่ ก่อนจะเปิดไฟล์สำหรับการส่งออก ให้ใช้ฟังก์ชัน Dir\$ ถ้าเรียกฟังก์ชัน Dir\$ ที่มีชื่อไฟล์เป็นพารามิเตอร์ ฟังก์ชันก็อาจส่งคืนสำเนาของชื่อไฟล์ (หมายถึงว่ามีไฟล์อยู่) หรือส่งคืนสคริงว่าง (หมายถึงว่าไม่มีไฟล์ที่มีชื่อตามที่กำหนดไว้) ก่อนโปรแกรมนี้แสดงฟังก์ชัน Dir\$

```
if Dir$("OEDIPUS.TXT") <> "" Then
    if inputBox("OEDIPUS.TXT already existx. Delete it?") <> "Yes"
        Then Exit Sub
    End if
End if
```

Open "OEDIPUS.TXT" For Output As # 15

หลังจากเสร็จสิ้นการทำงานกับไฟล์แล้วจะต้องเปิดไฟล์นั้น ข้อความสั่ง Close ของ Visual Basic มีวากยสัมพันธ์ดังต่อไปนี้

Close #fileNumber

หมายเลขไฟล์ในข้อความสั่ง Close จะต้องสอดคล้องกับหมายเลขที่กำหนดให้แก่ไฟล์ในข้อความสั่ง Open เมื่อปิดไฟล์จะเก็บข้อมูลทั้งหมดที่เขียนไปยังไฟล์ (ถ้ามี) และหลังจากนั้นโปรแกรมอื่น ๆ จะสามารถเข้าถึงไฟล์ได้โดยเสรี

การอ่านจากไฟล์ลำดับ

การอ่านจากไฟล์ลำดับนั้นทำได้ง่ายมาก ข้อความสั่งที่ใช้คือ Line Input # ซึ่งจะอ่านข้อความแต่ละบรรทัด เมื่อพิจารณาในวากยสัมพันธ์ของข้อความสั่ง จะเห็นว่าต้องระบุไฟล์ (ด้วยหมายเลข) และให้ตัวแปรสคริงที่จะใช้เก็บข้อความรับเข้า

Line Input #finenNumber, stringVariable

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ฟังก์ชัน EOF (ซึ่งแทนจุดสิ้นสุดไฟล์) จะทำให้ทราบว่าเมื่อใดที่ข้อมูลหมดสิ้นแล้วฟังก์ชันนี้จะรับอาร์กิวเมนต์เดียว ๆ (หมายเลขไฟล์) และส่งคืนค่า True ถ้าได้อ่านสารสนเทศทั้งหมดต่อไปนี้เป็นตัวอย่างของรหัสที่เปิด อ่าน และปิดไฟล์ลำดับ

Open FileVar For Input As # 1

Do While Not EOF(1)

Line input # 1, LineVar

Process LineVar 'User procedure to handle a line of text

Loop

Close # 1

รหัสนี้จะถือว่ามีความแปรปรวน FileVar และ LineVar อยู่ FileVar จะถูกกำหนดค่าเริ่มต้นให้เป็นชื่อของไฟล์ และรหัสจะอ่านบรรทัดแต่ละบรรทัดของไฟล์ข้อความเข้าไปไว้ในตัวแปร LineVar

การเขียนไปยังไฟล์ลำดับ

การเขียนไปยังไฟล์ลำดับสามารถทำได้ง่ายพอๆ กับการอ่านจากไฟล์ ข้อความสั่ง Print# จะเขียนไปยังไฟล์ลำดับด้วยแนวทางที่เหมือนกันกับที่กลวิธี Print เขียนไปยังตัวแบบฟอร์มวากยสัมพันธ์ของข้อความสั่ง Print# คือ

Print #fineNumber[, expression [{, !:} expression]...][, !:}]

ค่าของนิพจน์ที่แยกออกจากกันด้วยเซมิโคลอน (;) จะถูกเขียนโดยไม่มีช่องว่างอยู่ระหว่างค่าในขณะที่ค่าของนิพจน์ที่ถูกแยกออกจากกันด้วยจุลภาคจะถูกเขียนไปยังฟิลด์ปริ้นต์ (print field) ที่แยกต่างหากกันโดยมีความยาว 14 อักขระ อักขระปิดแคร์และอักขระป้อนบรรทัด จะถูกผนวกเข้ากับผลส่งออก เว้นแต่ว่าอักขระตัวสุดท้ายในข้อความสั่ง Print# จะเป็นจุลภาค CONFIG.SYS

การไม่สนใจขอบเขตของบรรทัด

แม้ว่าตามปกติจะใช้ข้อความสั่ง Print # เขียนบรรทัดเพียงบรรทัดเดียว หรือส่วนหนึ่งของบรรทัด แต่เราสามารถใช้อักขระสั่งนี้เขียนออกมาหลาย ๆ บรรทัดในครั้งเดียว เช่นที่ทำได้ในโปรแกรม Configurator ในทางตรงกันข้าม ข้อความสั่ง Line Input # จะถูกจำกัดให้ประมวลการรับเข้าเพียงทีละหนึ่งบรรทัด

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

อย่างไรก็ตาม เราสามารถใช้ฟังก์ชัน Input\$ ของ Visual Basic อ่านไฟล์รับเข้า (input file) ทีละหนึ่งอักขระ ฟังก์ชัน Input\$ จะส่งคืนสตริงที่อ่านจากไฟล์ ซึ่งรวมทั้งอักขระปิดแคว่และอักขระป้อนบรรทัด วากยสัมพันธ์เป็นดังต่อไปนี้

Input\$(charCount #fileNumber)

เราสามารถระบุให้อ่านตั้งแต่ 1 จนถึง 32,767 อักขระ (charCount) ภายในครั้งเดียว ถ้าไฟล์มีอักขระน้อยกว่าจำนวนที่ระบุไว้ Visual Basic จะก่อให้เกิดข้อความระบุนความผิดพลาด

Input past end of file

เราสามารถเขียนโปรแกรม Configurator ให้ใช้ฟังก์ชัน Input\$ ได้โดยง่าย ต่อไปนี้เป็นกระบวนการงานอีเวนต์ btnOpen__Click ซึ่งได้เขียนใหม่โดยใช้ Input\$

Sub btnOpen__Click()

Dim CLine As String

if ConfigFound Then

Beep

Exit Sub

End if

Open "C:\CONFIG.SYS" For Input As #1

txtContent.Text = Input\$(LOF(1), #1) 'Assume that file has fewer
'than 32,767 characters

Close # 1

ConfigFound = true

End Sub

กระบวนการนี้จะอ่านไฟล์ CONFIG.SYS ทั้งหมดภายในครั้งเดียว ซึ่งสามารถใช้ได้ก็เพราะว่า CONFIG.SYS เป็นไฟล์ที่มีขนาดค่อนข้างเล็กมาก ในกรณีของไฟล์ยาว ๆ จำเป็นต้องใช้ loop เพื่อตรวจสอบการสิ้นสุดของไฟล์ (EOR) สังเกตดูกระบวนการที่เขียนขึ้นใหม่ข้างต้นนี้จะเห็นว่าใช้ฟังก์ชัน LOF ซึ่งแทน length of file (ความยาวของไฟล์) ฟังก์ชันนี้จะรับหมายเลขไฟล์เป็นพารามิเตอร์และส่งคืนจำนวนไบต์ในไฟล์

ไฟล์เข้าถึงเชิงสุ่ม

นักเขียนโปรแกรมส่วนใหญ่มักจะประสบปัญหาสองประการกับการรับเข้าแบบลำดับ ปัญหาประการหนึ่งคือ มักจะไม่มีประสิทธิภาพที่จะแทนข้อมูลด้วยแนวทางที่ใช้ข้อความ เช่น ตามปกติจะสามารถเก็บจำนวน 42,1596 เอาไว้ในตัวแปรชนิด single ขนาด 4 ไบต์ แต่ถ้าเก็บจำนวนเป็นข้อความในไฟล์ลำดับแล้วจำเป็นต้องใช้ 7 ไบต์ ปัญหาอีกประการหนึ่งคือ ในบางครั้งจำเป็นต้องประมวลสารสนเทศด้วยแนวทางที่ไม่ใช่แบบลำดับ

ในตอนนี้เราย้อนกลับไปดู Bavarian Tufted Forest Weasels ในครั้งสุดท้ายที่เราพิจารณา โปรแกรมนี้เราได้สร้างฐานข้อมูลเพื่อบันทึกเกี่ยวกับวีเซล ข้อสมมติในเวลานั้นคือสารสนเทศทั้งหมดเกี่ยวกับวีเซลจะบรรจุอยู่ในตัวแปรโปรแกรม (อาร์เรย์) อย่างไรก็ตาม เมื่อใช้กลวิธีนี้ทำให้ในเวลาที่เราเปลี่ยนแปลงข้อมูลต้องเปลี่ยนแปลงโปรแกรมด้วย ซึ่งปัญหานี้จะเทียบเท่ากับการที่ต้องดัดแปลงโปรแกรมประมวลคำในทุก ๆ ครั้งที่จะพิมพ์เอกสารใหม่ทางแก้ทางหนึ่งคือ การเก็บข้อมูลไว้ในไฟล์ แต่กระนั้นจะต้องไม่เก็บในรูปแบบจำนวนเต็มจำนวนความเที่ยงสองเท่าเก็บเป็นความเที่ยงสองเท่า นอกจากนี้ควรจะเข้าถึงสถิติที่สำคัญของวีเซลได้ตลอดเวลา โดยไม่ต้องประมวลไฟล์ทั้งหมดเรียงลำดับต่อเนื่องจนกระทั่งไปพบกับสิ่งที่ต้องการ ในตัวอย่างถัดไปเราจะสร้างฐานข้อมูล Weasel เพื่อแสดงโปรแกรมที่ใช้ไฟล์เข้าถึงเชิงสุ่ม

การเปิดและการปิดไฟล์เข้าถึงเชิงสุ่ม

วากยสัมพันธ์สำหรับข้อความสั่ง Open ในเวลาที่ใช้กับไฟล์เข้าถึงเชิงสุ่มเป็นดังนี้

Open fileName For Random As fileNumber Len = recordLength

ข้อความสั่ง Open ในรุ่นนี้เราไม่จำเป็นต้องระบุภาวะการรับเข้าหรือการส่งออก เพราะเราสามารถอ่านจากและเขียนไปยังไฟล์เข้าถึงเชิงสุ่มโดยที่ไม่ต้องเปิดและปิดไฟล์ในแต่ละครั้งนอกจากนี้เราต้องกำหนดความยาวของเรคอร์ด ซึ่งเป็นจำนวนไบต์ที่จะอ่านหรือเขียนด้วยข้อความสั่ง Get หรือ Put (เป็นข้อความสั่งที่ใช้อ่านจากหรือเขียนไปยังไฟล์) เช่น ถ้าวางแผนจะใช้ไฟล์ที่ประกอบไปด้วยจำนวนเต็มทั้งหมด ความยาวของเรคอร์ดจะเป็น 2 ถ้าค่าทุก ๆ ค่าในไฟล์เป็นชนิด currency ความยาวของเรคอร์ดจะเป็น 8 ถ้าไม่ระบุความยาวของเรคอร์ด Visual Basic จะถือว่ามีความยาวของเรคอร์ดเป็น 128

ในการหาความยาวของเรคอร์ดสำหรับฐานข้อมูล Weasel ขั้นแรกเราจะต้องมาดูที่ข้อมูลที่วางแผนจะเขียนและอ่านได้นิยามชนิดข้อมูล BTFWeasel ให้เป็นดังต่อไปนี้

Type BTFWeasel

 Name As String

 Color As String

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Weight As Integer

Leangth As Integer

Birthdate As Bouble

TruftColor As String

TuftLength As Integer

End Type

สมมติว่าแต่ละเรคอร์ดมีลักษณะเช่นข้างต้นนี้อย่างแน่ชัด ดังนั้นก็อาจบวกลดฟิลด์ประกอบทั้งหมดเพื่อให้ได้ขนาดของเรคอร์ด แต่โชคไม่ดีนักมีอยู่สามฟิลด์เป็นสตริง และไม่มีทางทราบขนาดของสตริงเลย

ในการแก้ปัญหานี้ Visual Basic เปิดโอกาสให้ประกาศขนาดตายตัวสำหรับสตริง ถ้าใช้ความสามารถนี้ และกำหนดข้อจำกัดบางอย่างเกี่ยวกับข้อมูล ก็จะสามารถคำนวณความยาวของเรคอร์ดได้ เช่น ถ้าเราจำกัดชื่อของวีเซลให้มีความยาวสูงสุด 32 อักขระ และชื่อสีเป็น 12 อักขระ เราสามารถนิยามชนิด BTFWeasel ใหม่เป็นดังต่อไปนี้

Type BTFWeasel

Name As String * 32

Color As String * 12

Weight As Integer

Length As Integer

Birthdate As Double

TuftColor As String * 12

TuftLength As Integer

End Type

นิพจน์ String * 32 จัดสรรไบต์จำนวน 32 ไบต์อย่างตายตัวให้แก่ชื่อวีเซลแต่ละชื่อ ถ้าชื่อต้องการอักขระน้อยกว่า 32 ตัว Visual Basic จะอัดขยายชื่อด้วยช่องว่าง ชื่อที่ต้องการอักขระมากกว่า 32 ตัวจะถูกตัดให้สั้น ดังนั้นในตอนนี้อันความยาวรวมของเรคอร์ด BTFWeasel เป็น 70 ไบต์ (32 + 12 + 2 + 2 + 8 + 12 + 2)

การปิดไฟล์เข้าถึงเชิงสุ่มได้ง่ายมาก ใช้ข้อความสั่ง Close ในทำนองเดียวกันกับการปิดไฟล์ลำดับ เช่น ถ้าเปิดไฟล์เข้าถึงเชิงสุ่มในลักษณะ file # 1 ให้ใช้ข้อความสั่ง Close # 1 เพื่อปิดไฟล์

การอ่านจากและการเขียนไปยังไฟล์เข้าถึงเชิงสุ่ม

ข้อความสั่ง Get และ Put เป็นข้อความสั่งที่ใช้อ่านจากและเขียนไปยังไฟล์เข้าถึงเชิงสุ่ม วกยสัมพันธ์ของสองข้อความสั่งนี้เป็นดังต่อไปนี้

Get #fileNumber, [recordNumber], variable

Put #fileNumber, [recordNumber], variable

ข้อความสั่ง Get จะอ่านจากไฟล์และเก็บข้อมูลไว้ในตัวแปร ในขณะที่ข้อความสั่ง Put จะเขียนสิ่งที่บรรจุอยู่ในตัวแปรที่กำหนดไว้ไปยังไฟล์ ในแต่ละกรณีอาจเลือกระบุหมายเลขเรคอร์ดไว้ ถ้าไม่ได้กำหนดหมายเลขเรคอร์ดก็จะใช้ตำแหน่งลำดับถัดไป

ไฟล์เข้าถึงเชิงสุ่มมีลักษณะคล้ายกับอาร์เรย์ชนิดที่ผู้ใช้นิยามเป็นอย่างมาก ความแตกต่างสำคัญอยู่ตรงที่จะเก็บข้อมูลบนแผ่นดิสก์แทนที่จะเก็บอยู่ในหน่วยความจำ และดังนั้นข้อมูลจึงไม่หายไปเมื่อโปรแกรมยุติ (เรคอร์ดแต่ละเรคอร์ดบนแผ่นดิสก์จะถูกกำหนดหมายเลขดัชนีโดยเริ่มต้นที่เรคอร์ด 1 ในขณะที่อิลีเมนต์ของอาร์เรย์จะถูกกำหนดหมายเลขดัชนีโดยเริ่มต้นที่ 0) ตารางที่ 2.13 แสดงให้เห็นว่าจะเก็บเรคอร์ดบนแผ่นดิสก์อย่างไร

No.	ชื่อ	สี	น้ำหนัก	ความยาว	วันเกิด	สีขน	ขนยาว
1	Siegmune	น้ำตาล	18	11	09/13/91	น้ำตาล	6
2	Sieglinde	น้ำตาลอ่อน	14	10	10/29/91	ดำ	6

ตารางที่ 2.13
เรคอร์ดในไฟล์เข้าถึงเชิงสุ่ม

สมมติว่าในอนาคตจำเป็นต้องนิยามชนิด BTFWeasel ใหม่ โดยอาจจำเป็นต้องเพิ่มฟิลด์ใหม่หรืออาจเห็นว่าอักขระ 32 ตัวนั้นไม่เพียงพอสำหรับชื่อ ถ้าไม่ได้ใช้ชื่อค่าคงตัวแทนความยาวของเรคอร์ดแล้ว ก็อาจจำเป็นต้องตรวจสอบโปรแกรมทุก ๆ บรรทัด เพื่อหาจำนวน 70 และเปลี่ยนแปลงจำนวนนั้นให้ไปเป็นค่าใหม่ นอกจากนี้ยังจะต้องมั่นใจอีกว่า 70 แต่ละค่าเป็นการอ้างอิงไปยังขนาดของเรคอร์ดจริง ๆ ไม่ได้เป็นค่าอื่น ๆ การใช้ชื่อค่าคงตัวจะทำให้สามารถเพียงแต่เปลี่ยนค่าของค่าคงตัว และส่วนที่เหลือของโปรแกรมจะถูกปรับโดยทันทีเพื่อให้ทำงานได้กับขนาดของเรคอร์ดที่นิยามใหม่

ส่วนที่เหลือของการประกาศจะเป็นแบบเฉพาะที่ต่อแบบฟอร์ม ปัดวินโดว์รหัสสำหรับมอดูลรหัส และเปิดวินโดว์รหัสสำหรับแบบฟอร์ม เดิมบรรทัดต่อไปนี้จะเข้าไปในตอนการประกาศทั่วไป

Dim CurrentWeaselIx As Integer	'Number of current weasel
Dim WeaselCount As Integer	'Total number of weasel
Dim RecordModified As Integer	'Boolean flag

ตัวแปรเหล่านี้จะครอบครองหมายเลขดัชนี ซึ่งบ่งบอกว่าจะแสดงเรคอร์ดของวีเซลใดออกมาจะบันทึกจำนวนทั้งหมดของเรคอร์ดของวีเซลที่เก็บอยู่ในไฟล์ และจะติดตามว่าผู้ใช้ได้ดัดแปลงข้อมูลใด ๆ บนจอภาพหรือไม่ (ในโปรแกรมนี้ใช้คำย่อ Ix แทนที่คำว่า Index เพราะ Index เป็นชื่อของคุณสมบัติของ Visual Basic)

ในเวลาที่เราเริ่มใช้งานระบบ โปรแกรมควรจะอ่านแต่ละเรคอร์ดในไฟล์ และบรรจุชื่อของวีเซลทั้งหมดในลิสต์บ็อกซ์ หลังจากนั้นผู้ใช้สามารถเลือกชื่อ โปรแกรมจะตอบสนองด้วยการหยิบเอาเรคอร์ดทั้งหมดสำหรับวีเซลนั้น และแสดงเรคอร์ดนั้นออกมา ถ้าผู้ใช้เปลี่ยนแปลงแก้ไขเรคอร์ด ก็จะเขียนการเปลี่ยนแปลงที่ทำนั้นกลับไปยังไฟล์

ในการเขียนโปรแกรมที่จะทำงานได้ดีกับข้อมูลในปริมาณมาก ๆ ดังนั้นโปรแกรมนี้จะไม่อ่านเรคอร์ดทั้งหมดเข้าไปในอาร์เรย์ แต่จะทำงานทั้งหมดกับตัวแปรหน่วยความจำเป็นการแทนที่อาร์เรย์ไม่สามารถมีดัชนีได้มากกว่า 32,767 ค่านี้บรรจุวีเซลเป็นจำนวนมาก และที่ 70 ไบต์ต่อเรคอร์ดจะต้องการแรมมากกว่า 2 เมกะไบต์ ถ้าบันทึกเพียงทีละหนึ่งเรคอร์ดในหน่วยความจำก็จะช่วยให้สามารถลดปริมาณของหน่วยความจำที่ต้องการลงอย่างมาก โปรแกรมนี้จะให้ชื่อทั้งหมดบรรจุในหน่วยความจำพร้อมกัน แต่ถ้าชื่อโดยเฉลี่ยมีความยาวแปดหรือเก้าอักขระ สิ่งนี้ก็ไม่ควรที่จะเป็นปัญหา สมมติว่ามีไฟล์ข้อมูลปรากฏอยู่ในตำแหน่งที่กำหนดไว้บนแผ่นดิสก์แล้ว เราจะสามารถเขียนรูทีนกำหนดค่าเริ่มต้นดังต่อไปนี้ได้ (ในแอปพลิเคชันนี้ เราไม่ได้ใช้ฟังก์ชัน FileOpenDlg แต่เราจะถือเอาชื่อไฟล์และพาทที่ตายตัว สำหรับข้อมูล ชื่อพาท C:\VB จะระบุไครเรทอรีของ Visual Basic ที่อยู่บนระบบ ขอให้แน่ใจว่าได้ใช้ชื่อพาทที่ถูกต้องกับระบบที่ใช้)

Sub WeaselInit()

Dim Weasel As BTFWeasel, W As Integer

Open 'C:\VB\WEASEL.DAT' For Random As # 1 Len =

WEASEL_RECLEN

WeaselCount = LOF(1) / WEASEL_RECLEN

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

For W = 1 To WeaselCount

Get # 1, W, Weasel

cboName. AddItem Weasel.Name

Next W

End Sub

ข้อความสั่งนี้ทำหน้าที่เปิดหรือสร้างไฟล์ขึ้น ฟังก์ชัน LOF จะส่งคืนจำนวนทั้งหมดของไบต์ในไฟล์ซึ่งจะถูกหารด้วยขนาดของเรคอร์ด เพื่อให้ได้จำนวนของเรคอร์ดทั้งหมด หลังจากนั้นอ่านเรคอร์ดที่มีอยู่แต่ละเรคอร์ด และแทรกชื่อของวีเซลแต่ละชื่อเข้าไปในลิสต์บ็อกซ์

การจัดเตรียมการแสดงเรคอร์ดออกมานั้นทำได้ง่าย เพราะการเพียงแต่วก 1 เข้ากับดัชนีของชื่อในลิสต์บ็อกซ์จะตรงกันกับดัชนีของเรคอร์ดในไฟล์ ในรูปที่ที่สองจะขอให้ผู้ใช้ป้อนชื่อของวีเซลตัวใหม่ เมื่อกดปุ่มเมาส์บนปุ่ม Add ถ้าไม่ได้ป้อนชื่อเข้าไปจะไม่มีการเติมเรคอร์ดถ้าผู้ใช้ป้อนชื่อ กระบวนการจะเรียก NewWeasel เพื่อสร้างเรคอร์ด และผ่านค่าดัชนีที่ฟังก์ชันนี้ส่งคืนไปให้แก่ ShowWeasel โดยตรงที่จะได้แสดงเรคอร์ดใหม่ออกมา

การเพียงแต่ตั้งตัวบ่งชี้ในกระบวนการ Change แทนที่จะเรียกทูทีน UpdateCurrentWeasel โดยตรงเป็นวิธีการที่ดีที่สุด เพราะ Visual Basic จะเรียกกระบวนการ Change ในทุก ๆ ครั้งที่มีการเปลี่ยนแปลงเนื้อหาของเท็กซ์บ็อกซ์ เช่น ถ้าพิมพ์ข้อความ Black ในเท็กซ์บ็อกซ์ Color กระบวนการ Change จะถูกเรียกห้าครั้ง แต่แต่ละครั้งสำหรับการพิมพ์อักษรแต่ละตัวแต่การปรับเรคอร์ดไม่ควรเกิดขึ้นจนกว่าค่าในเท็กซ์บ็อกซ์จะเสร็จสมบูรณ์ ดังนั้นโปรแกรมจึงเรียกทูทีนปรับเรคอร์ดในกระบวนการอื่นคือ LostFocus เป็นการแทนที่ที่ LostFocus คืออะไร? ตัวควบคุมจอภาพใด ๆ อาจเป็น “จุดรวมของความสนใจ” ได้ในบางเวลา ในระหว่างการเฝ้าซิกซ์แอปพลิเคชัน Windows จะบันทึกจุดรวมความสนใจนี้ด้วยอีเวนต์ GotFocus ขึ้น จุดรวมความสนใจยังคงอยู่ในเท็กซ์บ็อกซ์ในขณะที่กำลังพิมพ์หลังจากนั้นถ้ากดแป้น Tab หรือกดปุ่มเมาส์บนตัวควบคุมอื่น ๆ บางตัว เท็กซ์บ็อกซ์จะรับอีเวนต์ LostFocus และตัวควบคุมใหม่จะได้รับ GotFocus ฉะนั้นเมื่อดำเนินการกระบวนการ LostFocus ของเท็กซ์บ็อกซ์ก็จะทราบได้ทันทีว่าผู้ใช้ได้เสร็จสิ้นการพิมพ์ และโปรแกรมสามารถเรียกทูทีนปรับเรคอร์ดได้อย่างปลอดภัย

การลบเรคอร์ด

แอปพลิเคชันนี้มีแบบจำลองเชิงตรรกะที่มีประสิทธิภาพ และทำความเข้าใจค่อนข้างง่ายแต่ความง่ายเช่นนั้นก็ส่งผลตอบแทนนั้นคือไม่มีหนทางง่าย ๆ ที่จะลบเรคอร์ด

สมมติว่าไฟล์ฐานข้อมูล WEASEL.DAT ประกอบไปด้วยเรคอร์ด สำหรับวีเซลที่ได้ตั้งชื่อให้เป็น A, B, C, และ D ตามที่แสดงในรูปที่ 2.50 การสร้างเรคอร์ดเหล่านี้จะเป็นการสร้างไฟล์ที่ประกอบไปด้วย 280 ไบต์ ($4 * 70$)

A
B
C
D

รูปที่ 2.50

ไฟล์ข้อมูลที่มีสี่เรคอร์ดสำหรับฐานข้อมูล Weasel

ในกรณีที่ต้องการลบเราไม่สามารถทำได้เพียงการกำจัดไบต์ออกจากตอนกลางของไฟล์และ “ขว้างทิ้งไป” เช่น ถ้าต้องการลบเรคอร์ดสำหรับวีเซล B ก็มีทางเลือกสองทางคือ คัดลอกเรคอร์ดทั้งหมดยกเว้น B ไปยังไฟล์ใหม่ หรือเขียนทับเรคอร์ด B ด้วยข้อมูลใหม่บางข้อมูลถ้าสร้างสำเนาขึ้น ก็สามารถลบไฟล์ตั้งต้นและตั้งชื่อสำเนาใหม่ด้วยชื่อไฟล์ตั้งต้น

ถ้าเลือกการเขียนทับเรคอร์ด B ด้วยข้อมูลอื่น ๆ เช่นเรคอร์ดที่ประกอบไปด้วยที่ว่างทั้งหมดในท้ายที่สุดลิสต์บ็อกซ์จะเต็มไปด้วยข้อมูลว่าง ๆ คุณอาจคิดว่ารูทิงกำหนดค่าเริ่มจะเพียงแต่ละเว้นเรคอร์ดใด ๆ ที่มีฟิลด์ชื่อว่างอยู่แต่โชคไม่ดีนักถ้าข้ามเรคอร์ดใด ๆ เรคอร์ดหนึ่งแอปพลิเคชันจะไม่สามารถจับคู่หน่วยข้อมูลของลิสต์บ็อกซ์กับเรคอร์ดที่เหมาะสม

แอปพลิเคชันดังเช่นที่เขียนล่าสุดต้องอาศัยการแปลงแบบ 1 ต่อ 1 (one-to-one mapping) ระหว่างชื่อในลิสต์บ็อกซ์และเรคอร์ดที่อยู่ในไฟล์(อนึ่งความต้องการเช่นนี้ยังหมายถึงการไม่สามารถตั้งคุณสมบัติ Sorted ของลิสต์บ็อกซ์ให้เป็น True เพราะการเรียงลำดับจะไปรบกวนสัมพันธภาพระหว่างค่าดัชนีของลิสต์บ็อกซ์กับตำแหน่งของเรคอร์ด)

ทางแก้อย่างไหนจะดีกว่ากัน? ถ้าทราบอย่างแน่ชัดว่าไฟล์ข้อมูลสำหรับแอปพลิเคชันจะมีขนาดเล็กเสมอ แน่่อนวิธีการคัดลอกและการตั้งชื่อใหม่ย่อมดีกว่า เราสามารถทำให้เกิดวิธีนี้ขึ้นโดยไม่ส่งผลกระทบต่อรหัสที่เขียนขึ้น และแอปพลิเคชันยังคงง่าย ๆ และทำความเข้าใจได้ง่าย อย่างไรก็ตาม ในโปรแกรมที่มีข้อมูลมาก ๆ การคัดลอกไฟล์ขนาดใหญ่จะทำให้เสียเวลามากและต้องการเนื้อที่ว่างของแผ่นดิสก์มากพอสำหรับทั้งไฟล์คัดลอกและไฟล์ตั้งต้นให้ปรากฏอยู่พร้อมกันในเวลาเดียวกัน ถ้าเนื้อที่ของแผ่นดิสก์ไม่พอ โปรแกรมก็อาจล้มเหลวได้ การสร้างแผนที่ทำให้โปรแกรมสามารถดำเนินงานได้รวดเร็วมากขึ้น แต่ทางแก้แบบนั้นจะเป็นการเพิ่มความซับซ้อน (หรือบางทีก็จุดบกพร่อง) และต้องเขียนรหัสที่มีอยู่บางส่วนใหม่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในฐานะของนักเขียนโปรแกรมมักจำเป็นต้องตัดสินใจในทำนองนี้อยู่เสมอ ๆ นักเขียนโปรแกรมจำเป็นต้องเข้าใจภารกิจของโปรแกรมและธรรมชาติของข้อมูลอย่างถ่องแท้ก่อนทำการตัดสินใจรหัสไม่ว่าจะปราศจากจุดบกพร่องมากน้อยเท่าใดก็ตาม ก็จะไม่มีความค่าเลยถ้าผู้ใช้ไม่สามารถอยู่กับการตัดสินใจที่ทำ

โปรแกรมที่เปลี่ยนแปลงแล้วใช้คุณสมบัติที่เรียกว่า ItemData ซึ่งเป็นคุณลักษณะแบบในตัวของลิสต์บ็อกซ์ ItemData เป็นอาร์เรย์ที่มีหน่วยข้อมูลเป็นจำนวนเต็มแบบยาวสำหรับอีลีเมนต์แต่ละตัวในลิสต์บ็อกซ์ เราใช้อาร์เรย์เก็บการแปลงส่ง นั่นคืออาร์เรย์จะครอบครองดัชนีของไฟล์บนแผ่นดิสก์ที่แท้จริงสำหรับหน่วยข้อมูลแต่ละหน่วยในฉบับดั้งเดิม ตัวแปร CurrentWeaselIx อาจหมายถึงเฉพาะดัชนีในลิสต์บ็อกซ์เท่านั้น ส่วนตำแหน่งไฟล์จะต้องได้มาจากนิพจน์ cboName.ItemData(CurrentWeaselIx)

การเปลี่ยนแปลงที่เกิดกับกระบวนการ NewWeasel, ShowWeasel และ UpdateCurrentWeasel ไม่ค่อยสำคัญนัก โดยเพียงแต่ทำให้มั่นใจว่าได้ใช้และบำรุงรักษาการแปลงส่งอย่างถูกต้อง การเปลี่ยนแปลงที่เกิดกับ WeaselInit ก่อนข้างตรงไปตรงมา ในขณะที่ฟังก์ชันอ่านแต่ละเรคอร์ดในไฟล์ ฟังก์ชันจะตรวจสอบว่าอักขระตัวแรกของฟิลด์ชื่อเป็นอักขระว่างหรือไม่ ถ้าเป็นอักขระว่างก็จะไม่สนใจเรคอร์ดนั้น ในขณะที่โปรแกรมอ่านเรคอร์ด โปรแกรมจะเขียนค่าที่เหมาะสมเข้าไปในอาร์เรย์แผนที่ ItemData คุณสมบัติ NewIndex จะบรรจุค่า ListIndex สำหรับชิ้นข้อมูลที่เติมเข้าไปล่าสุดที่สุด ด้วยเหตุที่เราไม่ต้องการสัมพันธ์ภาพแบบหนึ่งต่อหนึ่งระหว่างแผ่นดิสก์และลิสต์บ็อกซ์อีกต่อไป ดังนั้นจึงสามารถตั้งคุณสมบัติ Storted ของลิสต์บ็อกซ์ในโปรแกรมฉบับนี้ให้เป็น True

ไดนามิกลิงค์ไลบรารี

เราสามารถเข้าถึงฟังก์ชันที่สร้างเข้าไปใน Windows หรือสร้างเข้าไปในแอปพลิเคชันที่ดำเนินงานบน Windows ได้ถ้าฟังก์ชันเป็นส่วนหนึ่งของไดนามิกลิงค์ไลบรารี (DLL) ถ้าลองตรวจสอบดูไดเรกทอรี SYSTEM ของ Windows จะพบไฟล์จำนวนหนึ่งที่มีส่วนขยายชื่อไฟล์ DLL ไฟล์เหล่านี้บางไฟล์มาพร้อมกับ Windows ในขณะที่บางไฟล์ (เช่น VBRUN200.DLL) มาพร้อมกับแอปพลิเคชันที่ได้เติมเข้าไป ถ้าทราบบทนิยามของฟังก์ชันในไฟล์เหล่านี้ก็จะสามารถเรียกฟังก์ชันได้เกือบทุกฟังก์ชันจากแอปพลิเคชันของ Visual Basic

ในการเรียกฟังก์ชันเหล่านี้จาก Visual Basic จำเป็นต้องทราบสารสนเทศบางเทคนิคเกี่ยวกับชนิดข้อมูลการส่งผ่านพารามิเตอร์ และอีกหลาย ๆ อย่าง สารสนเทศเหล่านี้บางส่วนจะปรากฏอยู่ในคำบรรยายเกี่ยวกับข้อความสั่ง Declare ที่อยู่ในคู่มืออ้างอิง Visual Basic ส่วนสารสนเทศเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เกี่ยวกับฟังก์ชันที่เรียกได้แต่ละฟังก์ชันอย่างจำเพาะ ต้องลองตรวจดูในคู่มือการเขียนโปรแกรม สำหรับผลิตภัณฑ์ที่ให้ DLL

คำเตือน ข้อความสั่ง Declare เป็นข้อความสั่งที่มีประสิทธิภาพมาก ข้อความสั่งนี้ทำให้สามารถเข้าถึงฟังก์ชันระบบของ Windows ได้เกือบทุกฟังก์ชัน แต่การใช้ข้อความสั่งนี้จะต้องทำด้วยความระมัดระวัง ด้วยเหตุที่ Declare เรียกระบบโดยตรงดังนั้นจึงไม่ได้รับประโยชน์จากการตรวจหาความผิดพลาดของ Visual Basic ถ้าเกิดพิมพ์การประกาศผิดไป หรือเรียกฟังก์ชันระบบด้วยค่าที่ไม่ถูกต้องแล้ว ระบบอาจพังลงได้



เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

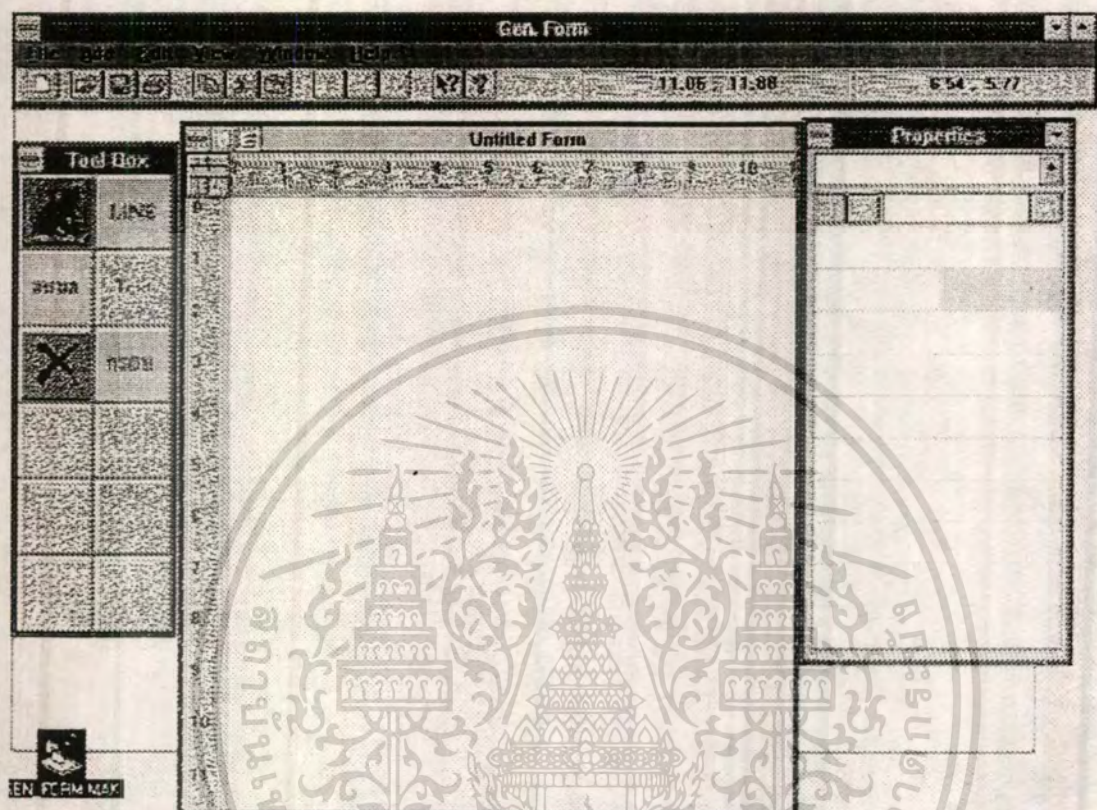
บทที่ 3 การสร้าง

การออกแบบเลือกให้โปรแกรมทำงานภายใต้สภาวะแวดล้อม วินโดวส์ เพราะว่าแนวโน้มโปรแกรมทั่วไปที่มีใช้กันจะพยายามพัฒนาให้มาทำงานภายใต้วินโดวส์กันทั้งนั้น การทำงานบนระบบวินโดวส์เป็นแบบ GUI (Graphic User Interface) มีความสวยงาม สื่อกับผู้ใช้ได้ง่าย รวมทั้งในตอนนี้มีเครื่องมือ (ภาษา) ที่ช่วยพัฒนาโปรแกรมบนวินโดวส์ให้เลือกมากมาย อีกทั้งมีความง่าย สะดวกและรวดเร็ว และโปรแกรมนี้จะต้องมีการติดต่อกับผู้ใช้ค่อนข้างมาก และยังต้องมีการใช้รูปแบบตัวอักษร (Font) ซึ่งวินโดวส์มีให้เลือกใช้มากมาย มีความสวยงาม อีกอย่างก็คือเรื่อง เครื่องพิมพ์ วินโดวส์จะมีไดรเวอร์ของเครื่องพิมพ์อยู่แล้ว ทำให้เราสามารถใส่เครื่องพิมพ์ได้หลายแบบ โดยที่เราไม่ต้องไปจัดการเอง

ต่อมาก็เลือกเครื่องมือในการพัฒนา (ภาษา) เลือกใช้ Visual Basic 3.0 เพราะเนื่องจากเคยใช้มาก่อนแล้วทำให้ไม่ต้องเสียเวลาศึกษามากนัก จากที่เคยได้ใช้มา ก็เห็นว่า สามารถพัฒนาโปรแกรมได้สะดวก รวดเร็วดี

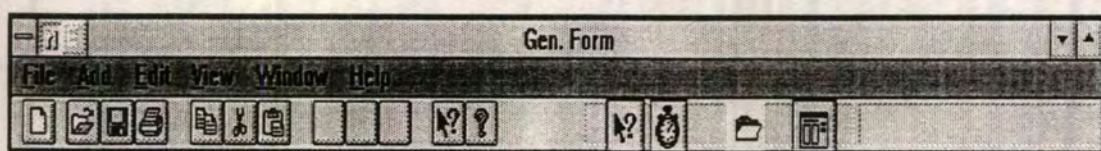
ต่อมาก็มาออกแบบโปรแกรม การทำงานและการติดต่อกับผู้ใช้ โดยได้แนวความคิดมาจากตัวโปรแกรม Visual Basic เอง เพราะการทำงานคล้าย ๆ กัน กล่าวคือต้องมีการออกแบบหน้าจอ (แบบฟอร์ม) ก่อนมีอุปกรณ์ให้เลือกใช้แล้วจากนั้นก็นำมาวางบนฟอร์ม มีการเปลี่ยนขนาดตำแหน่ง และคุณสมบัติอื่น ๆ ได้ จึงได้นำแนวคิดนี้มาสร้างเป็นโปรแกรมออกแบบและสร้างแบบฟอร์ม โดยแบ่งโปรแกรมออกเป็น 4 หน้าต่าง คือ หน้าต่างหลัก จะวางตัวอยู่บนสุดของหน้าจอ มีเมนูและทูลบาร์ เป็นหน้าต่างที่ควบคุมการทำงานหลักของโปรแกรม หน้าต่างต่อมาคือ หน้าต่างออกแบบ จะวางตัวอยู่กลางหน้าจอ เป็นหน้าต่างที่ใช้ในการนำอุปกรณ์ต่าง ๆ มาวางเพื่อสร้างแบบฟอร์ม จะมีไม้บรรทัดอยู่ทั้งแนวดิ่งและแนวนอนโดยสามารถเปลี่ยนหน่วยกลับไปมาระหว่างนิ้วกับเซนติเมตร โดยกดปุ่มที่บริเวณซ้ายมือบนได้คอนโทรลเมนู หน้าต่างออกแบบนี้จะสามารถเปิดทำงานได้เพียงหน้าต่างเดียว หน้าต่างต่อมาคือ หน้าต่างเครื่องมือ จะวางตัวอยู่ด้านซ้ายมือลักษณะจะมีปุ่มให้เลือกอยู่หลายปุ่มซึ่งหนึ่งปุ่มก็แทนอุปกรณ์หนึ่งอย่าง เมื่ออยากเลือกอุปกรณ์ตัวใดก็คลิกเมาส์ อุปกรณ์ดังกล่าวก็จะปรากฏที่หน้าต่างออกแบบทันที หน้าต่างสุดท้ายคือหน้าต่างคุณสมบัติมีหน้าที่แสดงคุณสมบัติของแต่ละอุปกรณ์และรับการแก้ไขคุณสมบัติเหล่านั้นจากผู้ใช้ เมื่อเรียกโปรแกรมขึ้นมาทำงานโปรแกรม จะเปิดหน้าต่างออกแบบว่าง ๆ มาให้โดยอัตโนมัติ ในการทำงานหลัก ๆ จะสามารถสั่งได้จากเมนูในหน้าต่างหลัก ในการเลือก อุปกรณ์เพื่อสร้างแบบฟอร์มสามารถเลือกได้จากทั้งเมนูและคลิกเมาส์ในหน้าต่างเครื่องมือ เมื่อนำอุปกรณ์ตัวใดไปวางบน

หน้าตาออกแบบ ที่หน้าตาคุณสมบัติก็จะแสดงคุณสมบัติของแต่ละอุปกรณ์นั้นที่ให้ผู้ใช้งานสามารถแก้ไขได้



รูปที่ 1 แสดงโปรแกรมขณะทำงาน

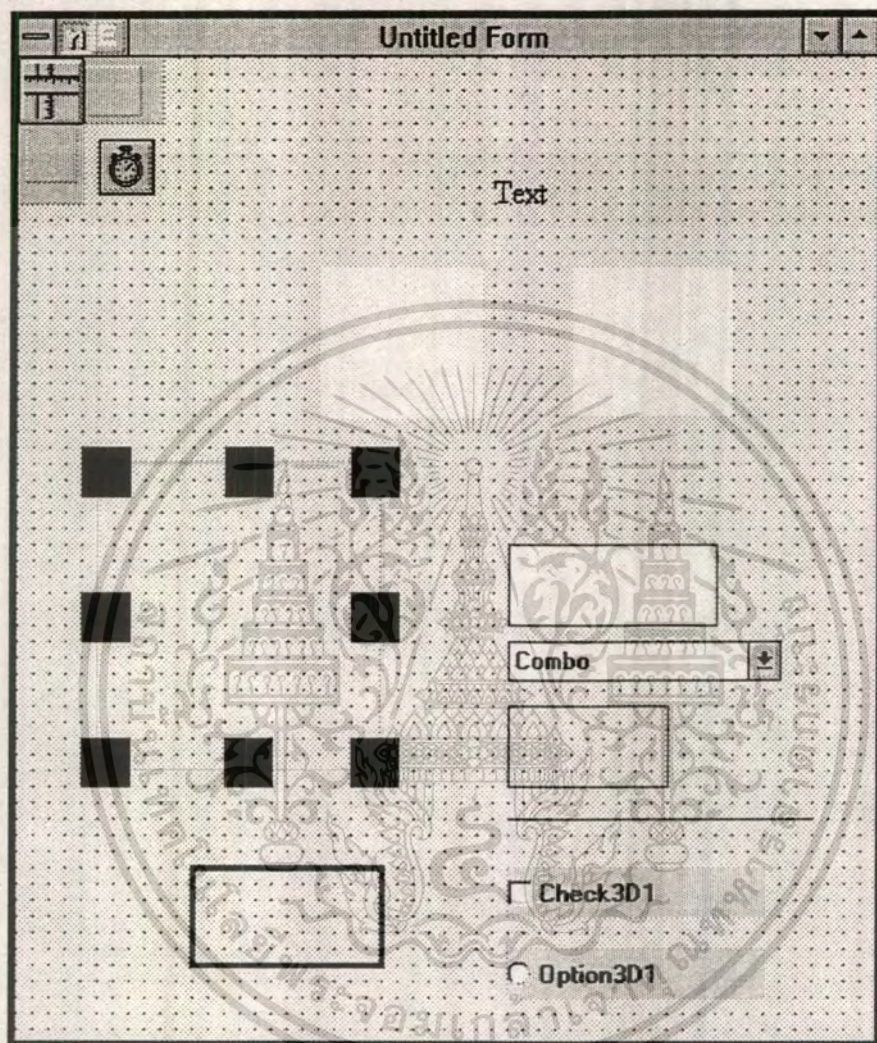
จากรูปที่ 1 เป็นผลการทำงานของโปรแกรมสร้างและออกแบบแบบฟอร์ม จะเห็นมีหน้าตาอยู่ 4 หน้าตา คราวนี้ลองมาดูหน้าตาแต่ละหน้าตาขณะออกแบบ โดยเริ่มจากหน้าตาหลักก่อน ก็จะประกอบไปด้วย ปุ่มที่ทำหน้าที่เป็น ToolBar วางอยู่บน Panel3D และก็มี Common Dialog และเมนู



รูปที่ 2 แสดงหน้าตาหลักขณะออกแบบสร้าง

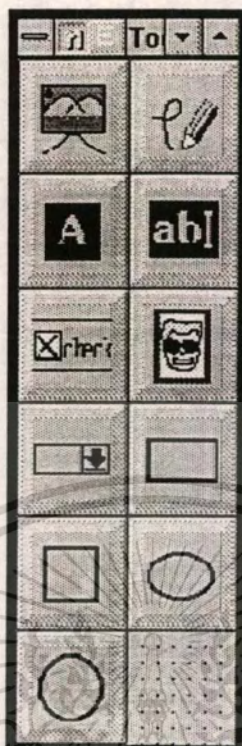
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ต่อมาก็เป็นหน้าต่างออกแบบที่จะประกอบไปด้วยอ็อบเจกต์มากมายที่สำคัญก็คืออ็อบเจกต์ที่จะใช้ในการสร้างแบบฟอร์ม ซึ่งเราใช้วิธีสร้างเอาไว้ก่อนอย่างละตัว จากนั้นก็ใช้วิธีสร้างอ็อบเจกต์ตัวนั้นขึ้นขณะโปรแกรมทำงานโดยให้เป็นอาร์เรย์

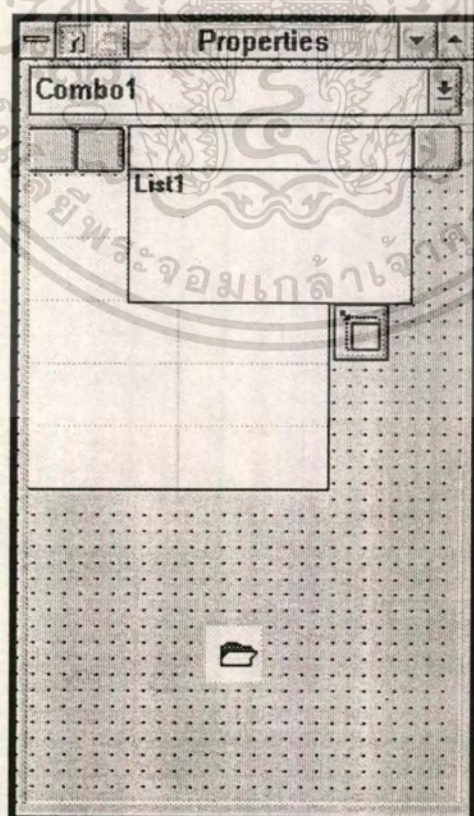


รูปที่ 3 แสดงหน้าต่างออกแบบขณะออกแบบ

ต่อมาคือหน้าต่างเครื่องมือซึ่งจะประกอบไปด้วยปุ่มเท่ากับจำนวนอุปกรณ์ที่ใช้สร้างแบบฟอร์ม



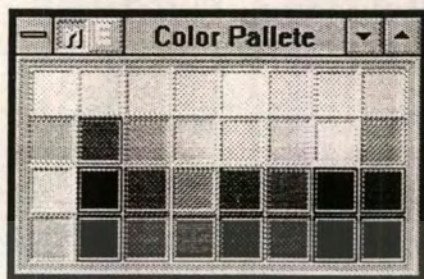
รูปที่ 4 แสดงหน้าต่างเครื่องมือขณะออกแบบ
ต่อมาก็คือหน้าต่างคุณสมบัติ ซึ่งอ็อบเจกต์ที่สำคัญก็ได้แก่ กริด



รูปที่ 5 แสดงหน้าต่างคุณสมบัติขณะออกแบบ

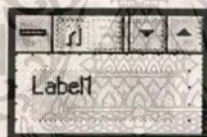
เอกสารนี้เป็นเอกสารที่สงวนลิขสิทธิ์สำหรับการใช้งานภายในเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

และยังมีอีก 2 หน้าต่างที่จะไม่แสดงตัวในเวลาปกติ ก็คือหน้าต่างเลือกสี ซึ่งจะแสดงก็ต่อเมื่อต้องการเปลี่ยนสีของอุปกรณ์



รูปที่ 6 แสดงหน้าต่างเลือกสีขณะออกแบบ

หน้าต่างนี้เป็นหน้าต่างที่จะแสดง Help Tip ต่าง ๆ ในขณะที่เราเลื่อนเมาส์ไปอยู่บนปุ่มต่าง ๆ ในระยะเวลาหนึ่ง



รูปที่ 7 แสดงหน้าต่างแสดง Tips ขณะออกแบบ

รายละเอียดการทำงานของโปรแกรม

Genform.mak

Color.frm เป็นฟอร์มที่ใช้แสดงหน้าต่างในการเลือกสีต่าง ๆ เพื่อกำหนดค่าให้กับคุณสมบัติของอ็อบเจกต์ต่าง ๆ โดยมีสีให้เลือก 32 สีโดยการคลิกเลือกสีที่ต้องการ

Form.frm เป็นฟอร์มที่ทำหน้าที่เป็นหน้าต่างออกแบบ ในขณะที่ทำงานจะแสดงไม้บรรทัดอยู่บนสุดและซ้ายสุดโดยที่จะแสดงได้ 2 หน่วยคือ เซนติเมตรและนิ้ว โดยถ้าต้องการจะเปลี่ยนหน่วยของไม้บรรทัดก็ทำได้โดยคลิกปุ่มที่อยู่ซ้ายมือบนสุดของฟอร์มนี้โดยจะทำงานสลับกันไปมาระหว่างเซนติเมตรและนิ้ว ฟอร์มนี้โดยปกติจะแสดงในขนาดเท่ากับขนาดที่เลือกตั้งแต่สร้างฟอร์ม แต่ถ้ามีขนาดใหญ่มากเกินไปก็จะแสดงในขนาดเท่าที่จะแสดงได้ดูพอเหมาะหรือในกรณีที่ผู้ใช้ปรับแต่งลด-เพิ่มขนาดของหน้าต่างเองแล้วมีขนาดเล็กกว่าขนาดจริง ก็จะแสดง Scroll Bar ขึ้นมา เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ในทางอื่นไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

เพื่อให้ผู้ใช้เลื่อนไปยังตำแหน่งในส่วนที่ไม่สามารถแสดงได้ โดยไม้บรรทัดก็จะเลื่อนไป แสดงในสัดส่วนกับการเลื่อน Scroll Bar ไป

Helptip.frm เป็นฟอร์มที่ใช้แสดงเป็น HelpTip โดยจะแสดงเมื่อผู้ใช้เลื่อนเมาส์ไปบนปุ่มต่าง ๆ ฟอร์มนี้ก็จะแสดงตัวขึ้นมาโดยจะเป็นการบอกหน้าที่ของปุ่มต่าง ๆ เหล่านั้น และก็จะหายไปเมื่อผู้ใช้เลื่อนเมาส์ออกหรือเลื่อนไปอยู่บนปุ่มอื่น

Linkdb.frm เป็นฟอร์มที่จะแสดงก็ต่อเมื่อผู้ใช้จะกรอกข้อความลงใน Text Box ในแบบฟอร์มที่ออกแบบ แล้วคลิกปุ่มที่อยู่ด้านขวามือของ Text Box ในหน้าต่างคุณสมบัติ โดยหน้าต่างนี้จะทำหน้าที่เชื่อมต่อกับฐานข้อมูล โดยเราจะต้องเลือกไฟล์ฐานข้อมูลก่อน จากนั้นก็เลือกตารางเลือกฟิลด์ และเลือกคอลัมน์ที่ต้องการจากนั้นก็คลิกปุ่ม Ok ข้อมูลนั้นก็จะปรากฏลงบน Text Box ในแบบฟอร์มบนหน้าต่างออกแบบ

Main.frm เป็นฟอร์มที่เป็นหน้าต่างหลัก โดยจะแสดงตัวอยู่บนสุด เป็นแนวยาวบนหน้าจอ จะมีเมนูและ Tool Bar

Newform.frm เป็นหน้าต่าง ที่ทำหน้าที่รับการระบุขนาดของแบบฟอร์มที่จะสร้างขึ้นใหม่ โดยจะมีขนาดให้เลือกตามมาตรฐานกระดาษที่ใช้กันอยู่โดยทั่วไปเช่น A4 เป็นต้น และยังสามารระบุขนาดเองได้ตามต้องการโดยสามารถเลือกป้อนได้ทั้งหน่วย มิลลิเมตร เซนติเมตร และนิ้ว

Property.frm เป็นหน้าต่างที่แสดงคุณสมบัติและรับการแก้ไขคุณสมบัติของอ็อบเจ็กต์แต่ละตัว โดยขึ้นอยู่กับว่าขณะนั้นอ็อบเจ็กต์ตัวใดแอคทีฟอยู่ คุณสมบัติของแต่ละอ็อบเจ็กต์อาจจะเหมือนหรือแตกต่างกันบ้าง การแก้ไขคุณสมบัติทำได้โดย ทำให้อ็อบเจ็กต์ที่ต้องการแอคทีฟ โดยคลิกไปบนอ็อบเจ็กต์นั้น แล้วจะปรากฏสี่เหลี่ยมสีดำเล็ก ๆ 8 อันรอบอ็อบเจ็กต์นั้น ยกเว้นกรณีที่เป็นเส้นจะมีแค่ 2 อัน จากนั้นที่หน้าต่างคุณสมบัติก็จะแสดงคุณสมบัติของอ็อบเจ็กต์ตัวนั้น จากนั้นเราก็เลือกคุณสมบัติที่ต้องการจะแก้ไขโดยคลิกที่กริดเซลล์นั้น ในการแก้ไขคุณสมบัติจะสามารถทำได้โดยการป้อนเข้าไปที่ Text Box โดยตรงหรือจะต้องเลือกจาก List Box หรือ จาก Dialog Box หรือจากกล่องสี่เหลี่ยมแล้วแต่คุณสมบัติข้อนั้น ๆ ของอ็อบเจ็กต์ตัวนั้น

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Tool.frm เป็นหน้าต่างที่มีปุ่มให้กดเลือกอ็อบเจ็กต์ที่จะนำไปสร้างแบบฟอร์ม โดยจะมีอ็อบเจ็กต์ให้เลือกทั้งหมด 11 ตัว

Formsize.bas เป็นไฟล์ที่ใช้ประกาศตัวแปรคุณสมบัติของอ็อบเจ็กต์แต่ละตัวที่ใช้ในการสร้างแบบฟอร์ม

Global.bas เป็นไฟล์ที่ใช้ประกาศตัวแปร Global ทั่วไป และเก็บฟังก์ชันที่ใช้เกี่ยวกับการปรับขนาดของอ็อบเจ็กต์และการแสดงว่าอ็อบเจ็กต์นั้น ๆ แอคทีฟ

Helptip.bas เป็นไฟล์ที่เก็บฟังก์ชันเกี่ยวกับการแสดง HelpTip

Cmdialog.vbx เป็นไลบรารีที่ใช้แสดง Dialog Box ต่าง ๆ

Grid.vbx เป็นไลบรารีที่ใช้ในการแสดงกริดที่ใช้ในหน้าต่างคุณสมบัติ

Opendlg.vbx เป็นไลบรารีที่ใช้แสดง Dialog Box Open และ Dialog Box Save

Sizer.vbx เป็นไลบรารีที่ใช้กำหนดขนาดของหน้าต่างว่าจะให้ย่อหรือขยายได้เท่าไร

Threed.vbx เป็นไลบรารีที่ใช้แสดง ปุ่ม Panel แบบ 3 มิติ

Toolbutn.vbx เป็นไลบรารีที่ใช้เป็น Tool Bar ที่หน้าต่าง Main

Color.frm

ColorBox_click(Index as integer) ColorBox เป็น Control ประเภท SSPanel โดยเป็น Array 32 ตัวโดยแต่ละตัวจะแสดงสีที่แตกต่างกัน เมื่อผู้ใช้คลิกเลือกลงบน ColorBox ใดก็จะนำ Control ประเภท SSPanel อีกตัวหนึ่งชื่อ Focus ออกมาแสดงทับบน ColorBox ตัวนั้น โดยมีสีเดียวกันแต่ละมีลักษณะนูนขึ้นมาเพื่อให้ผู้ใช้รู้สึกว่าได้คลิกลงไปจริง จากนั้นก็จะนำค่าสีที่ผู้ใช้เลือกไปแสดงที่ Text Box ใน Form4 หรือ หน้าต่าง Property โดยจะแสดงเป็นเลขฐานสิบหก

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Focus_click() Focus เป็น Control ประเภท SSPanel จะแสดงตัวที่ต่อเมื่อผู้ใช้คลิกลงบน ColorBox เหตุการณ์นี้จะเกิดเมื่อผู้ใช้คลิกข้ามสีเดิม โดยก็จะนำค่าสีที่ผู้ใช้เลือกไปแสดงที่ Text Box ใน Form4 หรือ หน้าต่าง Property โดยจะแสดงเป็นเลขฐานสิบหกเช่นกัน

Form_Load() เหตุการณ์นี้จะเกิดทุกครั้งเมื่อเริ่มโหลดฟอร์มนี้ขึ้นมา โดยจะเป็นการกำหนดตำแหน่งที่จะแสดงฟอร์มให้สัมพันธ์กับหน้าต่างคุณสมบัติ และกำหนดให้ Focus แสดงอยู่ที่ ColorBox ตำแหน่งแรก

Form.frm

Sub MoveEastBorder(X as Single, Y as Single) มีหน้าที่เปลี่ยนขนาดของ TrackingRectangle (สี่เหลี่ยมสีเทาเมื่อผู้ใช้ Drag อีอบเจ็คเพื่อย้ายตำแหน่ง) (สี่เหลี่ยมสีเทาเมื่อผู้ใช้ Drag อีอบเจ็คเพื่อย้ายตำแหน่ง) เมื่อผู้ใช้ปรับขนาดโดย Drag สี่เหลี่ยมด้านตะวันออก

Sub MoveNECorner (X As Single, Y As Single) มีหน้าที่เปลี่ยนขนาดของ TrackingRectangle (สี่เหลี่ยมสีเทาเมื่อผู้ใช้ Drag อีอบเจ็คเพื่อย้ายตำแหน่ง) เมื่อผู้ใช้ปรับขนาดโดย Drag สี่เหลี่ยมด้าน ตะวันออกเฉียงเหนือ

Sub MoveNorthBorder (X As Single, Y As Single) มีหน้าที่เปลี่ยนขนาดของ TrackingRectangle (สี่เหลี่ยมสีเทาเมื่อผู้ใช้ Drag อีอบเจ็คเพื่อย้ายตำแหน่ง) เมื่อผู้ใช้ปรับขนาดโดย Drag สี่เหลี่ยมด้านเหนือ

Sub MoveNWCorner (X As Single, Y As Single) มีหน้าที่เปลี่ยนขนาดของ TrackingRectangle (สี่เหลี่ยมสีเทาเมื่อผู้ใช้ Drag อีอบเจ็คเพื่อย้ายตำแหน่ง) เมื่อผู้ใช้ปรับขนาดโดย Drag สี่เหลี่ยมด้านตะวันตกเฉียงเหนือ

Sub MoveSECorner (X As Single, Y As Single) มีหน้าที่เปลี่ยนขนาดของ TrackingRectangle (สี่เหลี่ยมสีเทาเมื่อผู้ใช้ Drag อีอบเจ็คเพื่อย้ายตำแหน่ง) เมื่อผู้ใช้ปรับขนาดโดย Drag สี่เหลี่ยมด้านตะวันออกเฉียงใต้

Sub MoveSouthBorder (X As Single, Y As Single) มีหน้าที่เปลี่ยนขนาดของ TrackingRectangle (สี่เหลี่ยมสีเทาเมื่อผู้ใช้ Drag อีอบเจ็กเพื่อย้ายตำแหน่ง) เมื่อผู้ใช้ปรับขนาด โดย Drag สี่เหลี่ยมด้านใต้

Sub MoveSWCorner (X As Single, Y As Single) มีหน้าที่เปลี่ยนขนาดของ TrackingRectangle (สี่เหลี่ยมสีเทาเมื่อผู้ใช้ Drag อีอบเจ็กเพื่อย้ายตำแหน่ง) เมื่อผู้ใช้ปรับขนาด โดย Drag สี่เหลี่ยมด้านตะวันตกเฉียงใต้

Sub MoveWestBorder (X As Single, Y As Single) มีหน้าที่เปลี่ยนขนาดของ TrackingRectangle (สี่เหลี่ยมสีเทาเมื่อผู้ใช้ Drag อีอบเจ็กเพื่อย้ายตำแหน่ง) เมื่อผู้ใช้ปรับขนาด โดย Drag สี่เหลี่ยมด้านตะวันตก

Sub MoveXY1 (X As Single, Y As Single) มีหน้าที่เปลี่ยนขนาดของ Line4 (เส้นสีเทา เวลาคลิกลงบนอีอบเจ็กเส้น) เมื่อผู้ใช้ปรับขนาดโดย Drag สี่เหลี่ยมด้านซ้าย

Sub MoveXY2 (X As Single, Y As Single) มีหน้าที่เปลี่ยนขนาดของ Line4 (เส้นสีเทา เวลาคลิกลงบนอีอบเจ็กเส้น) เมื่อผู้ใช้ปรับขนาดโดย Drag สี่เหลี่ยมด้านขวา

Sub UpdateSize (l As Single, h As Single, t As Single, w As Single, num1 As Integer, num2 As Integer, num3 As Integer, num4 As Integer) มีหน้าที่ทำการ Update ตัวเลขแสดงตำแหน่งและขนาดของอีอบเจ็กหลังผู้ใช้ทำการปรับเปลี่ยนโดยใช้เมาส์

Sub Check_Click (Index As Integer, Value As Integer)

Sub Check_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Check_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Check_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single) ทั้ง 4 Sub ข้างบนจะจัดการ เกี่ยวกับอีอบเจ็ก Check Box ทั้งการ เปลี่ยนขนาด ตำแหน่ง ถ้าคลิกก็จะทำการแสดงคุณสมบัติของอีอบเจ็กที่แอคทีฟ

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Sub Form_DblClick () ถ้าไม่มีอ็อบเจ็กต์ใดแอคทีฟอยู่ และหน้าต่างคุณสมบัติเปิดอยู่ก็ให้ปิด แต่ถ้ามีอ็อบเจ็กต์แอคทีฟอยู่ก็ให้แสดงหน้าต่างคุณสมบัติออกมา

Sub Form_KeyUp (KeyCode As Integer, Shift As Integer) มีหน้าที่คอยตรวจสอบว่าผู้ใช้กดคีย์ Del หรือไม่ ถ้ากดแล้วก็ให้ลบอ็อบเจ็กต์ที่แอคทีฟอยู่ออกไป

Sub Form_Load () เป็นการกำหนดค่าเริ่มต้น ตัวแปรต่าง ๆ เช่น ขนาด ตำแหน่ง ของอ็อบเจ็กต์ต่าง ๆ ที่จะแสดง

Sub Form_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single) จะคอยจัดการเกี่ยวกับอ็อบเจ็กต์ที่ไม่มี Even เป็นของตัวเอง เช่น กรอบ เส้น และรูปร่างต่าง ๆ เพื่อให้เหมือนว่ามีเหตุการณ์ Mouse Click

Sub Form_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single) มีหน้าที่จัดการเกี่ยวกับการเคลื่อนย้ายตำแหน่งอ็อบเจ็กต์ต่าง ๆ บนฟอร์ม

Sub Form_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single) มีหน้าที่ Update ขนาดและตำแหน่งของอ็อบเจ็กต์หลังผู้ใช้ปรับ

Sub Form_Resize () มีหน้าที่ปรับขนาดของไม้บรรทัดและสเกลบาร์ให้สัมพันธ์กับขนาดและตำแหน่งของฟอร์ม

Sub Handles_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single) เป็นการกำหนดตำแหน่งและขนาดสี่เหลี่ยมสีเทาเมื่อผู้ใช้จะปรับขนาดของอ็อบเจ็กต์

Sub HandLine_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single) เป็นการกำหนดตำแหน่งและขนาดเส้นสีเทาเมื่อผู้ใช้จะปรับขนาดของอ็อบเจ็กต์เส้น

Sub HelpTip_Timer () ทำหน้าที่แสดง HelpTip เมื่อผู้ใช้เคลื่อน เมาส์ไปบนไม้บรรทัด และปุ่มเปลี่ยนหน่วย

Sub HScroll1_Change () ทำหน้าที่ปรับตำแหน่งของไม้บรรทัดและอ็อบเจ็กต์ต่าง ๆ ที่อยู่บนหน้าต่างออกแบบ

Sub Image1_Click (Index As Integer)

Sub Image1_DblClick (Index As Integer)

Sub Image1_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Image1_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Image1_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single) ทั้ง 5 Sub ข้างบนจะจัดการ เกี่ยวกับอ็อบเจ็กต์ Image ทั้งการเปลี่ยนขนาด ตำแหน่ง ถ้าคลิกก็จะทำการแสดงคุณสมบัติของอ็อบเจ็กต์ที่แอกทีฟ

Sub Label1_Click (Index As Integer)

Sub Label1_DblClick (Index As Integer)

Sub Label1_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Label1_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Label1_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single) ทั้ง 5 Sub ข้างบนจะจัดการ เกี่ยวกับอ็อบเจ็กต์ Label ทั้งการเปลี่ยนขนาด ตำแหน่ง ถ้าคลิกก็จะทำการแสดงคุณสมบัติของอ็อบเจ็กต์ที่แอกทีฟ

Sub Line1_Click (Index As Integer)

Sub Line1_DblClick (Index As Integer)

Sub Line1_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Sub Line1_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Line1_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Line1_Paint (Index As Integer)

Sub Line1_Resize (Index As Integer) ทั้ง 7 Sub ข้างบนจะจัดการ เกี่ยวกับอ็อบเจ็กต์ Line ทั้งการเปลี่ยนขนาด ตำแหน่ง ถ้าคลิกก็จะทำการแสดงคุณสมบัติของอ็อบเจ็กต์ที่แอกทีฟ

Sub Picture1_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Picture1_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Picture1_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single) มีหน้าที่จัดการเกี่ยวกับไม้บรรทัดแนวนอน ซึ่งถ้ามีการคลิกเมาส์แล้วจะมีเส้นประแนวตั้ง เพื่อใช้เป็นแนวเทียบตำแหน่ง

Sub Picture2_MouseDown (Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Picture2_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Picture2_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single) มีหน้าที่จัดการเกี่ยวกับไม้บรรทัดแนวตั้ง ซึ่งถ้ามีการคลิกเมาส์แล้วจะมีเส้นประแนวนอน เพื่อใช้เป็นแนวเทียบตำแหน่ง

Sub ScaleRule_Click () มีหน้าที่จัดการเกี่ยวกับการเปลี่ยนหน่วยของไม้บรรทัด และเปลี่ยนหน่วยการแสดงผลทั้งหมดเป็นหน่วยปัจจุบัน

Sub Text_Click (Index As Integer)

Sub Text_DblClick (Index As Integer)

Sub Text_MouseDown (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Text_MouseMove (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single)

Sub Text_MouseUp (Index As Integer, Button As Integer, Shift As Integer, X As Single, Y As Single) ทั้ง 5 Sub ข้างบนจะจัดการ เกี่ยวกับอ็อบเจ็กต์ Text Box ทั้งการเปลี่ยนขนาด ตำแหน่ง ถ้าคลิกก็จะทำการแสดงคุณสมบัติของอ็อบเจ็กต์ที่แอกทีฟ

Sub VScroll1_Change () ทำหน้าที่ปรับตำแหน่งของไม้บรรทัดและอ็อบเจ็กต์ต่าง ๆ ที่อยู่บนหน้าต่างออกแบบ

Helptip.frm

Sub Form_Load () เป็นการกำหนดตำแหน่งของ Label ที่จะแสดง และกำหนดสีพื้นหลังของทั้งฟอร์มและลาเบลเป็นสีเหลือง

Linkdb.frm

Sub Combo1_Click () เป็นการเลือกฟิลด์จาก Database ที่เลือกโดยจะต้องเลือกตารางมาก่อน เมื่อเลือกฟิลด์ได้แล้วก็อ่านว่ามีคอลัมน์ใดบ้างไปกำหนดค่าให้ Combo2

Sub Combo3_Click () เป็นการเลือกตารางจาก Database ที่เลือกโดยจะต้องเปิด Database Name มาก่อน เมื่อเลือกตารางได้แล้วก็อ่านว่ามีฟิลด์ใดบ้างไปกำหนดค่าให้ Combo1

Sub Command1_Click () เป็นการเลือก Database name โดยเป็นไฟล์ของ Access

Sub Command2_Click () หลังจากเลือก Columnที่ต้องการจาก Combo2 แล้วถ้าข้อมูลถูกต้องแล้ว ก็จะเป็นการนำข้อมูลกำหนดให้กับ อ็อบเจ็กต์ Text Box แล้วปิดฟอร์มไป

Sub Command3_Click () เป็นการยกเลิกการเลือกทั้งหมดและปิดฟอร์มไป

Main.frm

Sub Form_Load () เป็นการกำหนดค่าเริ่มต้นของตัวแปรต่าง ๆ และเรียกให้โหลดฟอร์มอื่น ตามมา

Sub Form_Resize () มีหน้าที่ตรวจสอบว่าฟอร์มนี้มีสถานะเป็นไอคอนหรือไม่ ถ้าใช่ต้องซ่อนฟอร์มอื่น ๆ ที่เปิดอยู่ให้หมด ทำเป็นเหมือนว่าถ้าเรา Minimize Main Form ลงไปแล้ว ฟอร์มลูกก็ควรจะถูกปิดไป แต่ถ้าผู้ใช้ดับเบิลคลิกบนไอคอนจนฟอร์มขยายกลับมาเป็นขนาดเดิมก็จะแสดงฟอร์มลูกออกมา

Sub Form_Unload (Cancel As Integer) เมื่อปิดฟอร์มหลักก็ต้องปิดฟอร์มอื่น ๆ ตามด้วย

Sub MenFileExit_Click () เป็นการออกจากโปรแกรม

Sub MenFileNew_Click () เปิดฟอร์มใหม่

Sub MenFileOpen_Click () เปิดฟอร์มที่มีอยู่แล้ว

Sub MenFilePmAll_Click () เป็นการสั่งพิมพ์ทั้งหมด

Sub MenFilePmFrm_Click () เป็นการสั่งพิมพ์เฉพาะฟอร์ม

Sub MenFilePmTxt_Click () เป็นการสั่งเฉพาะ Text

Sub MenFileSvAs_Click () เป็นการสั่งให้บันทึกเปลี่ยนชื่อฟอร์มที่สร้าง

Sub MenWinProp_Click () เป็นการสั่งให้แสดง/ซ่อนหน้าต่างคุณสมบัติ

Sub MenWinTool_Click () เป็นการสั่งให้แสดง/ซ่อนหน้าต่างเครื่องมือ

Sub ToolOpen_Click () เปิดฟอร์มที่มีอยู่แล้ว

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Sub ToolPm_Click () เป็นการสั่งพิมพ์

Sub ToolSave_Click () เป็นการสั่งให้บันทึกฟอร์มที่สร้างขึ้น

Newform.frm

Sub Cancel_Button_Click () เป็นการยกเลิกการเลือก และปิดฟอร์มไป

Sub Combo1_Click () โดย Combo1 นี้จะเก็บขนาดของกระดาษขนาดต่างๆ ไว้ให้เลือก
เช่น A4

Sub Combo2_Click () เป็นการเลือกหน่วยที่จะแสดงของความกว้าง

Sub Combo3_Click () เป็นการเลือกหน่วยที่จะแสดงของความสูง

Sub Form_Load () เป็นการกำหนดค่าเริ่มต้นของ Combo Box ทั้ง 3 ตัว

Sub Form_Resize () เป็นการสั่งให้แสดงฟอร์มอยู่ตรงกลางหน้าจอ

Sub Ok_Button_Click () ยืนยันและยอมรับขนาดที่เลือกว่าถูกต้อง

Property.frm

Sub Add_Font () ทำหน้าที่อ่านชื่อรูปแบบตัวอักษร (Font Name) จากระบบแล้วใส่ไว้ใน
List1

Function OnlyNum (key As Integer) ทำหน้าที่ตรวจสอบว่าตัวอักษรที่ผู้ใช้ป้อนเข้ามา
เป็นเฉพาะตัวเลขหรือไม่

Sub Combo1_Click () โดย Combo1 นี้จะเก็บรายชื่อของอ็อบเจกต์ทั้งหมดที่มีอยู่ใน
หน้าต่างออกแบบ เมื่อเราเลือกไปที่อ็อบเจกต์ตัวใดก็จะทำให้ตัวนั้นแอคทีฟ

Sub Command3D1_Click () โดย Command3D1 นี้จะคลิกได้ก็ต่อเมื่อเราเลือกที่จะเปลี่ยนคุณสมบัติของอ็อบเจกต์นั้น ๆ โดยที่คุณสมบัตินั้นจะต้องเป็นแบบให้เลือก คือเปลี่ยนได้เฉพาะที่ให้เลือก โดยสังเกตจากเมื่อเรากดคลิกบนตารางคุณสมบัติแต่ละอัน ถ้าจุดคำ 3 จุดบนปุ่มมีสีคำก็แสดงว่าสามารถคลิกได้ แต่ถ้าเป็นสีเทาที่แสดงว่าคลิกไม่ได้ต้องป้อนโดยตรงใน Text Box ในเหตุการณ์นี้มีหน้าที่ตรวจสอบว่าคุณสมบัติข้อนั้นของอ็อบเจกต์ที่แอกทีฟอยู่ขณะนั้น มีรายการอะไรให้เลือก หรือว่าต้องเปิดไฟล์อะไรหรือเป็นการเลือกสีหรือไม่ ถ้ามีรายการให้เลือก ก็จะแสดง List Box ชื่อ List1 พร้อมรายการให้เลือก ถ้าเกี่ยวกับไฟล์ก็จะแสดง ไดอะล็อกบ็อกซ์ หรือถ้าเกี่ยวกับสีก็จะแสดง Form5 ขึ้นมา เป็นต้น

Sub Form_GotFocus () มีหน้าที่กำหนดเซลล์ที่จะแอกทีฟตอนโหลดฟอร์มนี้ขึ้นมา ซึ่งก็คือเซลล์ในตำแหน่งบนสุด(แถวแรก)

Sub Form_Load () มีหน้าที่กำหนดค่าเริ่มต้น ขนาด ตำแหน่ง ของกริด และเคลียร์ List Box ชื่อ List1 ในครั้งแรก

Sub Form_LostFocus () มีหน้าที่ตรวจสอบว่า Form5 ยังแสดงตัวอยู่หรือไม่ ถ้ายังแสดงตัวอยู่ก็ให้ซ่อนตัวเอง

Sub Form_Resize () มีหน้าที่ปรับเปลี่ยนขนาดและตำแหน่งของอ็อบเจกต์ตัวอื่น ๆ ในฟอร์มให้สัมพันธ์กับขนาดของฟอร์มในปัจจุบันเช่น ขนาดของกริด

Sub Form_Unload (Cancel As Integer) มีหน้าที่ตรวจสอบว่า Form5 ยังแสดงตัวอยู่หรือไม่ ถ้ายังแสดงตัวอยู่ก็ให้ซ่อนตัวเอง

Sub Grid1_Click () มีหน้าที่นำค่าของคุณสมบัติของอ็อบเจกต์ที่แอกทีฟอยู่ไปแสดงที่ Text Box ชื่อ Text1 และตรวจสอบว่าคุณสมบัติข้อนั้น ๆ มีรายการอะไรให้เลือกหรือเกี่ยวข้องกับสี ฐานข้อมูลหรือไฟล์หรือไม่ ถ้าเกี่ยวข้องก็ให้แสดงรูปจุด 3 จุดเป็นสีคำและ Enable ปุ่ม Command3D1 แต่ถ้าไม่เกี่ยวข้องก็ให้แสดงเป็นสีเทา Disable ไม่ให้กดได้

Sub Grid1_KeyDown (KeyCode As Integer, Shift As Integer) มีหน้าที่คอยตรวจสอบว่าคีย์ที่กดเข้ามาเป็นคีย์ลูกศรซ้ายหรือขวาหรือไม่ ถ้าใช่ก็กำหนดค่าคีย์ใหม่ให้เป็น 0 เสมือนว่าไม่ได้กดคีย์เลย

Sub Grid1_KeyUp (KeyCode As Integer, Shift As Integer) มีหน้าที่คอยตรวจสอบว่าคีย์ที่กดเข้ามาเป็นคีย์ลูกศรขึ้นหรือลงหรือไม่ ถ้าใช่ก็เช็ลที่แอคทีฟทั้งแถวตามลูกศรขึ้นหรือลง และนำค่าของคุณสมบัติของอ็อบเจกต์ที่แอคทีฟอยู่ไปแสดงที่ Text Box ชื่อ Text1

Sub Grid1_MouseMove (Button As Integer, Shift As Integer, X As Single, Y As Single) ถ้าผู้ใช้คลิกเมาส์ค้างไว้แล้วลากเมาส์ไปมาบนกริดก็จะเช็ทให้แถวที่เมาส์ลากผ่านแอคทีฟทั้งแถว

Sub List1_Click () เมื่อผู้ใช้คลิกปุ่ม Command3D1 แล้วก็จะมี List Box ชื่อ List1 แสดงออกมาซึ่งขึ้นอยู่กับว่าเป็นอ็อบเจกต์ใดและคุณสมบัติข้อไหน อาจจะมีการแสดงไม่เหมือนกัน และเมื่อผู้ใช้เลือกรายการใดรายการหนึ่ง ก็มีหน้าที่นำคุณสมบัติที่เลือกไปกำหนดให้กับอ็อบเจกต์ที่เลือก และนำค่าของคุณสมบัติของอ็อบเจกต์ที่แอคทีฟอยู่ไปแสดงที่ Text Box ชื่อ Text1 ด้วย

Sub List1_MouseUp (Button As Integer, Shift As Integer, X As Single, Y As Single) มีหน้าที่สั่งให้ List1 ซ่อนตัวเอง คือหลังจากผู้ใช้เลือกรายการใดรายการหนึ่งแล้วก็ให้ปิดตัวเองไป

Sub Text1_GotFocus () เมื่อเกิดเหตุการณ์นี้ ตัวอักษรที่อยู่ใน Text1 ก็จะถูกเลือก (Select) ระบายสีทั้งหมด

Sub Text1_KeyPress (KeyAscii As Integer) ถ้ามีอ็อบเจกต์ใดแอคทีฟอยู่ แล้วผู้ใช้แก้ไขคุณสมบัติอันหนึ่งโดยการพิมพ์เข้าไปโดยตรงที่ Text1 แล้วกดคีย์ Enter ก็จะนำคุณสมบัตินั้นมาตรวจสอบว่าถูกต้องหรือไม่ เช่น ตำแหน่งซ้าย ก็จะต้องป้อนเข้ามาแต่ตัวเลขเท่านั้น หรือเกี่ยวกับสี ก็ต้องเป็นเลขฐานสิบหก เป็นต้น ถ้าถูกต้องแล้วก็จะนำค่านี้นำมากำหนดให้กับอ็อบเจกต์

Tool.frm

Sub Form_Load () กำหนดค่าเริ่มต้นตัวแปร

Sub HelpTip_Timer () มีหน้าที่แสดง HelpTip โดยตรวจสอบเมาส์ว่าเคลื่อนที่อยู่นบนปุ่มใดปุ่มหนึ่งหรือไม่ ถ้าใช่ก็แสดง HelpTip ของปุ่มนั้น ๆ ออกมา

Sub PushBorder_Click (Value As Integer) มีหน้าที่วางอ็อบเจ็กต์ กรอบ ลงบนหน้าต่าง ออกแบบ โดยมีการกำหนดคุณสมบัติมาตรฐานไว้ด้วย

Sub PushChk_Click (Value As Integer) มีหน้าที่วางอ็อบเจ็กต์ Check Box ลงบนหน้าต่าง ออกแบบ โดยมีการกำหนดคุณสมบัติมาตรฐานไว้ด้วย

Sub PushLabel_Click (Value As Integer) มีหน้าที่วางอ็อบเจ็กต์ ลาเบล ลงบนหน้าต่าง ออกแบบ โดยมีการกำหนดคุณสมบัติมาตรฐานไว้ด้วย

Sub PushLine_Click (Value As Integer) มีหน้าที่วางอ็อบเจ็กต์ เส้น ลงบนหน้าต่าง ออกแบบ โดยมีการกำหนดคุณสมบัติมาตรฐานไว้ด้วย

Sub PushPic_Click (Value As Integer) มีหน้าที่วางอ็อบเจ็กต์ รูปภาพ ลงบนหน้าต่าง ออกแบบ โดยมีการกำหนดคุณสมบัติมาตรฐานไว้ด้วย

Sub PushShape_Click (Index As Integer, Value As Integer) มีหน้าที่วางอ็อบเจ็กต์ รูปทรงต่าง ๆ คือ วงกลม วงรี สี่เหลี่ยมจัตุรัส และสี่เหลี่ยมผืนผ้า ลงบนหน้าต่าง ออกแบบ โดยมีการกำหนดคุณสมบัติมาตรฐานไว้ด้วย

Sub PushText_Click (Value As Integer) มีหน้าที่วางอ็อบเจ็กต์ Text Box ลงบนหน้าต่าง ออกแบบ โดยมีการกำหนดคุณสมบัติมาตรฐานไว้ด้วย

Formsize.bas

Type PointApi ประกอบด้วยตัวแปร x และ y เป็น Integer ทั้งคู่

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

Type MinMaxInfo ประกอบด้วย ตัวแปรประเภท PointApi เอาไว้เก็บค่าของตำแหน่ง ขนาด ทั้งสูงสุดและต่ำสุด

Type PropLabel เป็นตัวแปรที่ใช้เก็บคุณสมบัติของอ็อบเจ็กต์ Label

Type PropLine เป็นตัวแปรที่ใช้เก็บคุณสมบัติของอ็อบเจ็กต์ Line

Type PropImage เป็นตัวแปรที่ใช้เก็บคุณสมบัติของอ็อบเจ็กต์ Image

Type PropBorder เป็นตัวแปรที่ใช้เก็บคุณสมบัติของอ็อบเจ็กต์ Border

Type PropCheck เป็นตัวแปรที่ใช้เก็บคุณสมบัติของอ็อบเจ็กต์ Check Box

Type PropCombo เป็นตัวแปรที่ใช้เก็บคุณสมบัติของอ็อบเจ็กต์ Combo Box

Type PropShape เป็นตัวแปรที่ใช้เก็บคุณสมบัติของอ็อบเจ็กต์ Shape

Type PropText เป็นตัวแปรที่ใช้เก็บคุณสมบัติของอ็อบเจ็กต์ Text Box

Global.bas

Sub BDrawSelectionRectangle () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 8 อันที่บริเวณขอบของอ็อบเจ็กต์ กรอบ

Sub CbDrawSelectionRectangle () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 8 อันที่บริเวณขอบของอ็อบเจ็กต์ Combo Box

Sub CDrawSelectionRectangle () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 8 อันที่บริเวณขอบของอ็อบเจ็กต์ Check Box

Sub HideHandles () มีหน้าที่ซ่อนสี่เหลี่ยมสีดำทั้ง 8 อันแสดงอยู่รอบอ็อบเจ็กต์

Sub HideHandLine () มีหน้าที่ซ่อนสี่เหลี่ยมสีดำ 2 อันแสดงอยู่รอบอ็อบเจ็กต์เส้นเท่านั้น

Sub IDrawSelectionRectangle () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 8 อันที่บริเวณขอบของอ็อบเจ็กต์ รูปภาพ

Sub L1DrawSelectionRectangle () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 2 อันที่บริเวณปลายของอ็อบเจ็กต์ เส้น

Sub LbDrawSelectionRectangle () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 8 อันที่บริเวณขอบของอ็อบเจ็กต์ ลาเบล

Sub SDrawSelectionRectangle () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 8 อันที่บริเวณขอบของอ็อบเจ็กต์ รูปร่างต่าง ๆ

Sub ShowHandles () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 8 อันที่บริเวณขอบของอ็อบเจ็กต์

Sub ShowHandles2 () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 2 อันที่บริเวณขอบของอ็อบเจ็กต์

Sub TDrawSelectionRectangle () มีหน้าที่แสดงสี่เหลี่ยมสีดำ 8 อันที่บริเวณขอบของอ็อบเจ็กต์ Text Box

Helptip.bas

Type PointType ประกอบด้วยตัวแปร x และ y เป็น Integer ทั้งคู่

Sub ShowHelpTip (TipText As String) มีหน้าที่กำหนดตำแหน่งและขนาดของ HelpTip

บทที่ 4 การทดลองและผลการทดลอง

การทดลองเกี่ยวกับ ภาพลักษณ์ (Image)

ชุดอุปกรณ์ทิศทาง

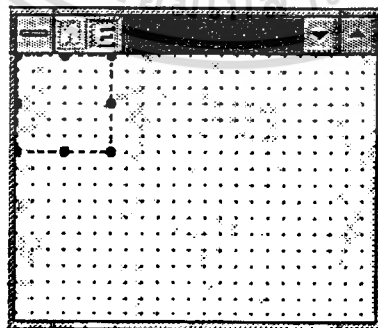
โครงการ (Project) ง่าย ๆ ต่อไปนี้ให้คุณได้ทดลองใช้ภาพลักษณ์จำนวนหนึ่งสร้างชุดอุปกรณ์ไอคอนที่มีลักษณะคล้ายกับทุลบาร์ของ Visual Basic โปรแกรมนี้จะทำงานในทำนองเดียวกันนี้ถ้าคุณใช้บ็อกซ์ภาพแทนที่ตัวควบคุมภาพลักษณ์ อย่างไรก็ตาม จำไว้อย่างหนึ่งว่าบ็อกซ์ภาพมีประสิทธิภาพน้อยกว่าเมื่อการแสดงผลเป็นแบบสถิต (Static)

1. สร้างโครงการใหม่ใน Visual Basic โดยการเลือกคำสั่ง New Project จากเมนู File
2. ตั้งคุณสมบัติของแบบฟอร์มตามที่แสดงอยู่ในที่นี้ (ไอคอน[none] แสดงถึงว่าค่ากำหนด

ควรจะเป็นสตริงว่าง)

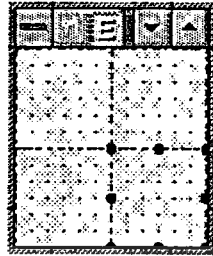
คุณสมบัติ	ค่ากำหนด
BorderStyle	1-Fixed Single
Caption	[none]
MaxButton	False
MinButton	False
Name	Directions

3. กดปุ่มเมาส์บนเครื่องมือสร้างภาพลักษณ์ในทุลบาร์ และลากพอยน์เตอร์ในแบบฟอร์มเพื่อสร้างตัวควบคุมภาพลักษณ์ขนาดเล็กตรงมุมด้านบนซ้ายของแบบฟอร์ม



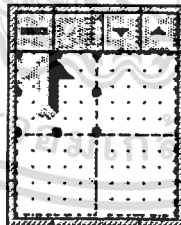
รูปที่ 4.1

4. สร้างตัวควบคุมภาพลักษณะเพิ่มเติมอีกสามตัวที่มีขนาดเท่ากัน และจัดตัวควบคุมเหล่านั้นในรูปสี่เหลี่ยมจัตุรัสขนาด 2*2 เปลี่ยนขนาดขอบเขตของฟอร์มวินโดว์โดยอาศัยเมาส์เพื่อให้พอเหมาะกับตัวควบคุมภาพลักษณะสี่ตัว ในตอนนี้แบบฟอร์มควรมีลักษณะเช่นดังนี้



รูปที่ 4.2

5. กดปุ่มเมาส์บนตัวควบคุมภาพลักษณะตรงมุมด้านบนซ้ายของแบบฟอร์ม (Image1) เพื่อเลือกตัวควบคุมนั้น ในวินโดว์ Properties ให้เลือกคุณสมบัติ Picture สำหรับ Image1 จกรายการคุณสมบัติ หลังจากนั้นกดปุ่มเมาส์บนปุ่มไขว้ปลาที่อยู่ทางด้านข้างของเซตดิงบ็อกซ์เพื่อเปิดไดอะล็อกบ็อกซ์ Load Picture ที่คุณใช้ในตัวอย่างก่อนหน้านี้ ในไดอะล็อกบ็อกซ์ ให้ตั้งชนิดของไฟล์เพื่อแสดงไอคอน (ที่มีส่วนขยาย ICO) ในไดเรกทอรีของ ARROWS ของไดเรกทอรี ICONS ที่มากับ Visual Basic คุณจะพบไฟล์ ARWO2UP.ICO เมื่อคุณเลือกไฟล์นั้น ไอคอนที่ไฟล์บรรจุอยู่จะปรากฏอยู่ในตัวควบคุมภาพลักษณะที่เลือกไว้



รูปที่ 4.3

6. กดปุ่มเมาส์สองครั้งบนภาพลักษณะ และนิยามกระบวนการงานดังต่อไปนี้ในวินโดว์รหัส

```
Sub Image1_Click()
```

```
    Debug.Print "Selected up"
```

```
End Sub
```

7. ทำกระบวนการอย่างเดียวกันนี้ซ้ำสำหรับตัวควบคุมภาพลักษณ์อื่น ๆ อีกสามตัว แล้ว
บรรจุไฟล์ ARW02DN.ICO, ARW02LT.ICO และ ARW02RT.ICO และนิยามกระบวนการงาน
อีเวนต์ Click อีกสามกระบวนการงานดังต่อไปนี้

```
Sub Image2_Click()
    Debug.Print "Selected down"
End Sub
```

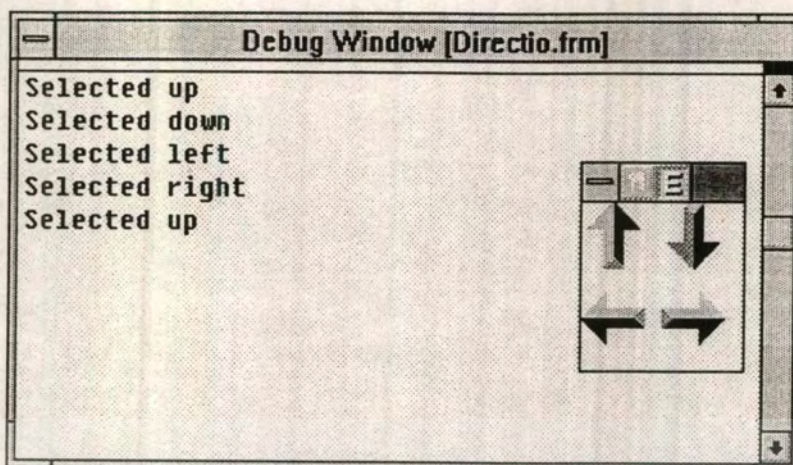
```
Sub Image3_Click()
    Debug.Print "Selected left"
End Sub
```

```
Sub Image4_Click()
    Debug.Print "Selected right"
End Sub
```

8. ในการแยกไอคอนออกจากกัน คุณอาจใส่ขอบไว้รอบ ๆ ไอคอน กดปุ่มเมาส์บนภาพ
ลักษณะที่อยู่ตรงมุมบนซ้ายเพื่อเลือกภาพลักษณ์นั้น และหลังจากนั้นกดปุ่มแป้น Ctrl ค้างไว้ใน
ขณะเดียวกันกับที่กดปุ่มเมาส์บนภาพลักษณ์ที่เหลืออีกสามภาพเพื่อที่จะเลือกภาพลักษณ์ทั้งสี่ภาพ
ในเวลาเดียวกัน ในวินโดว์ Properties เลือกคุณสมบัติ BorderStyle จากรายการคุณสมบัติและตั้ง
ให้เป็น 1-Fixed Single

เมื่อคุณดำเนินการงานแอปพลิเคชันนี้ (โดยการกด F5) แบบฟอร์มของคุณจะเป็นชุดอุปกรณ์
“Directions” เมื่อคุณลองกดปุ่มเมาส์บนไอคอน คุณจะเห็นผลลัพธ์เช่นที่แสดงอยู่ในรูปที่ 4.4

เมื่อรวมรหัสในการทำงานเดียวกันนี้เข้าไว้ในแอปพลิเคชันขนาดใหญ่ คุณก็จะสามารถสร้าง
ชุดอุปกรณ์ของไอคอนที่เหมาะสมกับโปรแกรมที่คุณกำลังออกแบบ ทูลบาร์ของ visual Basic ซึ่ง
ใช้ในระหว่างโหมดการออกแบบเป็นตัวอย่างหนึ่งของชุดอุปกรณ์ชนิดนี้



รูปที่ 4.4

การใช้ชุดอุปกรณ์ Directions

การทดลองเกี่ยวกับ Text Box

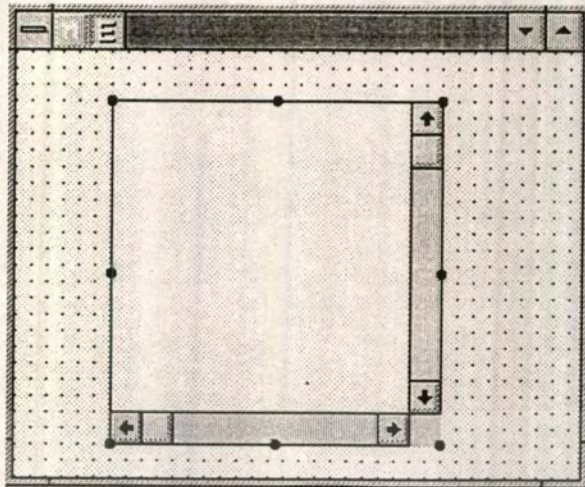
แอปพลิเคชัน NoteEdit

ที่ผ่านมาคุณได้ใช้เท็กซ์บ็อกซ์สำหรับข้อความบรรทัดเดียวมา แล้ว ในตอนนี้ลองมาดูว่าเท็กซ์บ็อกซ์จะประพฤติปฏิบัติในภาวะหลายบรรทัดอย่างไร

เริ่มต้นด้วยการเปิดโครงการใหม่หนึ่งโครงการ ตั้งคุณสมบัติ Caption และ Name ของแบบฟอร์มให้เป็น NoteEdit สร้างเท็กซ์บ็อกซ์ในแบบฟอร์ม และตั้งคุณสมบัติให้เป็นดังต่อไปนี้

คุณสมบัติ	ค่ากำหนด
MultiLine	True
Name	txtBox
ScrollBars	3-Both
Text	[none]

ในตอนนี้แบบฟอร์ม NoteEdit ควรจะมีลักษณะเช่นข้างล่างนี้



รูปที่ 4.5

แท็กชื่อบ็อกซ์จะจัดการเกี่ยวกับการรับข้อความเข้า และการแก้ไขข้อความโดยที่คุณไม่จำเป็นต้องไปยุ่งเกี่ยวกับการเขียนรหัส เมื่อคุณลงดำเนินงานแอปพลิเคชันนี้ คุณสามารถจะป้อนข้อความ แก้ไขข้อความ และตัด คัดลอก และปะ โดยอาศัยเป็นลัดมาตรฐานของ Windows นั่นคือ Ctrl-X, Ctrl-C และ Ctrl-V

ถ้าต้องการปรับปรุงแอปพลิเคชันให้ดียิ่งขึ้น เช่น คุณอาจต้องการทำแท็กชื่อบ็อกซ์ให้บรรจุอยู่ในฟอร์มวินโดว์เต็มพอดี สามารถทำได้โดยแก้ไขกระบวนการงาน *Resize* ของแบบฟอร์มเช่นที่แสดงในที่นี้

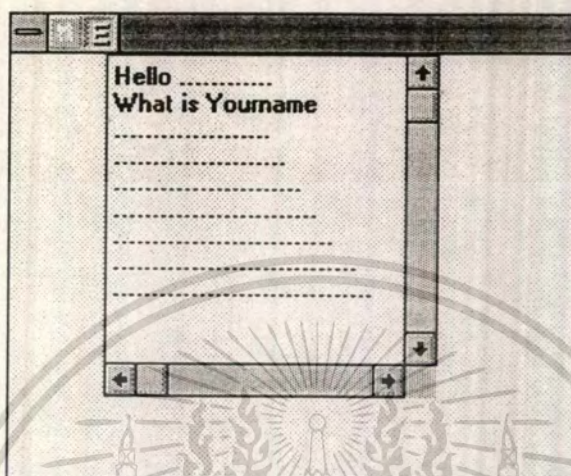
```
Sub Form_Resize()  
    txtBox.Top = 0  
    txtBox.Left = 0  
    txtBox.Width = ScaleWidth  
    txtBox.Height = ScaleHeight
```

End Sub

การเอ็กซีคิวต์กระบวนการนี้จะเกิดขึ้นในทุก ๆ ครั้งที่มีการเปลี่ยนขนาดแบบฟอร์ม รวมทั้งในเวลาที่เราแสดงแบบฟอร์มออกมาเป็นครั้งแรก การตั้งคุณสมบัติ *Top* และ *Left* ของแท็กชื่อบ็อกซ์ให้เป็น 0 ทำให้มันใจได้ว่ามุมด้านบนซ้ายนั้นพอดีกับส่วนบนสุดของแบบฟอร์มตามที่แสดงไว้ในรูปที่ 4.6 คุณสมบัตินี้ *Width* และ *Height* ของแท็กชื่อบ็อกซ์จะถูกตั้งให้พอดีกันกับคุณสมบัติ *ScaleWidth* และ *ScaleHeight* ของแบบฟอร์ม (ขนาดเต็มที่ของพื้นที่แสดงผลของแบบฟอร์ม) คุณอาจเขียนสองบรรทัดนี้เป็น `txtBox.Width = NoteEdit.ScaleWidth` และ `txtBox.Height`

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

= `NoteEdit.ScaleHeight` อย่างไรก็ตาม ด้วยเหตุที่จะเอ็กซีคิวต์รหัสในบริบทของแบบฟอร์มเสมอ ดังนั้นคุณจึงไม่จำเป็นต้องใช้ชื่อของแบบฟอร์ม นอกจากนี้ถ้าคุณเคยเปลี่ยนชื่อของแบบฟอร์ม คุณก็ไม่จำเป็นต้องย้อนกลับและไปแก้ไขรหัส



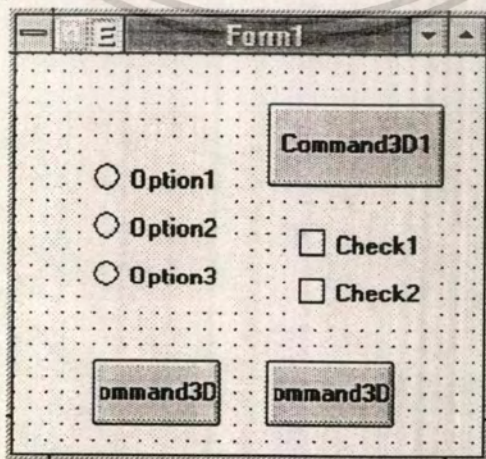
รูปที่ 4.6

การดำเนินงานแอปพลิเคชัน NoteEdit ที่เปลี่ยนขนาดแล้ว

การทดลองเกี่ยวกับ Option , Check Box and Command Button

แอปพลิเคชัน Game

โครงการนี้นำเสนอชนิดของ “โลมหน้า” ที่คุณอาจพบบนเกมวิดีโอ เริ่มต้นด้วยการเปิดโครงการใหม่ขึ้นหนึ่งโครงการ สร้างปุ่มคำสั่งขึ้นสามปุ่ม ปุ่มออพชั่นสี่ปุ่ม และไดอะล็อกบ็อกซ์สองบ็อกซ์โดยจัดเรียงตามรูปแบบดังนี้



รูปที่ 4.7

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตั้งคุณสมบัติของออบเจกต์ให้เป็นไปตามที่แสดงไว้ในตารางที่ 4.1 หลังจากที่คุณได้ตั้งคุณสมบัติแล้วแบบฟอร์มของคุณควรจะเป็นเช่นที่แสดงในรูปที่ 4.8

ออบเจกต์	คุณสมบัติ	ค่ากำหนด
Option1	Caption	1 Player
	Value	True
Option2	Caption	2 Players
Option3	Caption	3 Players
Option4	Caption	4 Players
Check1	Caption	Sound
Check2	Caption	Bonus Points
Command1	Caption	Coin Return
Command2	Caption	Start
	Default	True
Command3	Cancel	True
	Caption	Quit

ตารางที่ 4.1
การตั้งคุณสมบัติสำหรับแอปพลิเคชัน Game

The screenshot shows a standard Windows application window titled 'Form1'. Inside the window, there is a grid of controls. On the left, there are four radio buttons labeled '1Player', '2Player', '3Player', and '4Player'. To the right of these are two checkboxes labeled 'Sound' and 'Bonus Point'. At the bottom left are two buttons labeled 'Start' and 'Quit'. At the top right is a button labeled 'Coin Return'.

รูปที่ 4.8

อินเตอร์เฟซของ Game

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในขั้นสุดท้ายแก้ไขกระบวนการ Click สำหรับปุ่ม Command2 และ Command3 เพื่อให้รวมเอารหัสนี้เข้าไป

```
Sub Command2_Click()  
    MsgBox "Game Over"
```

```
End Sub
```

```
Sub Command3_Click()
```

```
    End
```

```
End Sub
```

ลองดำเนินการโปรแกรมดูสักสองสามครั้งเพื่อดูว่าโปรแกรมนั้นจะประพฤติปฏิบัติอย่างไรพยายามกดแป้น Enter และ Esc เพื่อดูว่าคำสั่งใดที่มีการเอ็กซีกิวต์

ถ้าคุณต้องการแสดงสถานะของปุ่มออปชันและเช็คบ็อกซ์ในวินโดว์ Debug คุณสามารถเปลี่ยนกระบวนการ Click สำหรับ Command2 ให้เป็นเช่นที่แสดงในที่นี้

```
Sub Command2_Click()  
    Debug.Print Option1.Value  
    Debug.Print Option2.Value  
    Debug.Print Option3.Value  
    Debug.Print Option4.Value  
    Debug.Print Check1.Value  
    Debug.Print Check2.Value
```

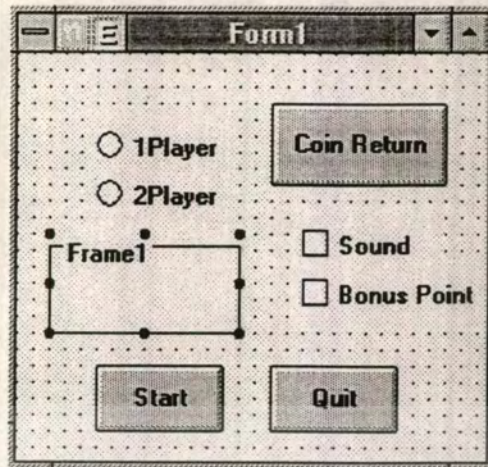
```
End Sub
```

การทดลองเกี่ยวกับ Frame

การดัดแปลงแอปพลิเคชัน Game

ด้วยเหตุที่กรอบไม่ได้มีอะไรน่าสนใจมากนัก ดังนั้นในตอนนี้จึงจะเพียงแต่ดัดแปลงโครงการของ Game จากตอนก่อนหน้านี้ แทนที่จะสร้างแอปพลิเคชันใหม่ขึ้นมา

1. ในฟอร์มวินโดว์ของแอปพลิเคชัน Game ให้ลบปุ่มออปชันที่ติดเลเบล 3 Players และ 4 Players ตรงตำแหน่งนี้ให้วาดกรอบคล้ายกับที่แสดงในที่นี้



รูปที่ 4.9

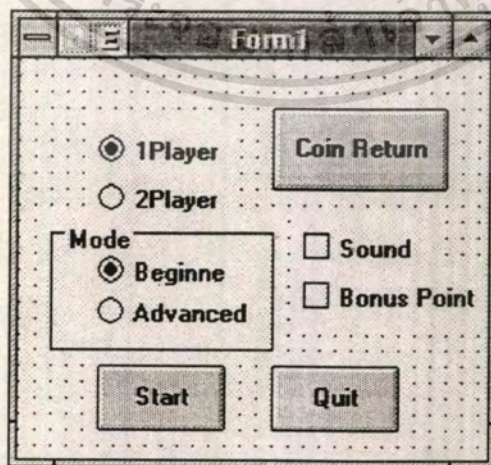
2. วางปุ่มออพชันใหม่สองปุ่มเข้าไปในกรอบ คุณจะต้องวาดปุ่มเหล่านี้เข้าไปภายในกรอบ ในกรณีนี้คุณไม่สามารถกดปุ่มเมาส์สองครั้งบนเครื่องมือสร้างปุ่มออพชันเพื่อสร้างปุ่มขึ้นและจากนั้นลากเข้าไปไว้ในกรอบ

3. ตั้งคุณสมบัติ Caption ของกรอบให้เป็น Mode

4. ตั้งคุณสมบัติ Value ของปุ่ม Option3 ให้เป็น True

5. ตั้งคุณสมบัติ Caption ของปุ่ม Option3 ให้เป็น Beginner และคุณสมบัติ Caption ของปุ่ม Option4 ให้เป็น Advanced

เมื่อทำเสร็จแล้วผลลัพธ์ที่ได้ควรเป็นเช่นที่แสดงในรูปที่ 4.10 เมื่อคุณลองดำเนินการ โปรแกรมจะสังเกตเห็นว่าปุ่มออพชันที่อยู่ภายในกรอบจะดำเนินการตามคำสั่งแยกออกจากปุ่มที่อยู่ภายนอกกรอบ



รูปที่ 4.10

อินเตอร์เฟซของ Game ที่ได้รับการออกแบบใหม่

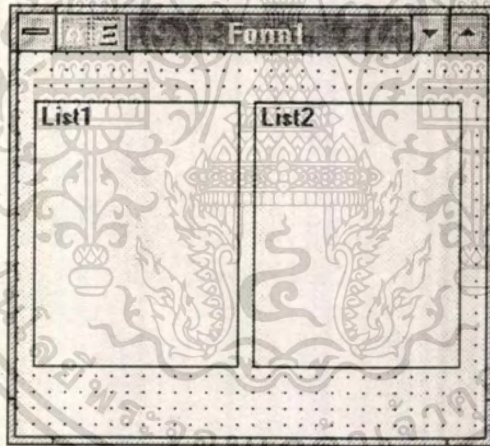
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทดลองเกี่ยวกับ List Box

แอปพลิเคชัน Cheese

โครงการตัวอย่างนี้จะถ่ายโอนชั้นรายการระหว่างลิสต์บ็อกซ์สองบ็อกซ์ ชั้นรายการในบ็อกซ์หนึ่งจะเรียงลำดับตามลำดับอักษร ส่วนชั้นรายการในบ็อกซ์อีกบ็อกซ์หนึ่งจะเรียงตามลำดับที่เพิ่มขึ้นรายการนั้นเข้าไป

เริ่มต้นด้วยการสร้างโครงการใหม่ขึ้นมาหนึ่งโครงการบนแบบฟอร์ม ให้วาดลิสต์บ็อกซ์สองบ็อกซ์ตามที่แสดงอยู่ในรูปที่ 4.11 และตั้งคุณสมบัติ Sorted ของลิสต์บ็อกซ์ที่อยู่ทางด้านขวา (List2) ให้เป็น True ส่วนที่เหลือของแอปพลิเคชันจะสร้างขึ้นโดยอาศัยรหัสที่แสดงไว้ในรูปที่ 4.12 กระบวนการ `form_Load` จะกำหนดค่าเริ่มต้นของลิสต์บ็อกซ์ ส่วนกระบวนการ `List DbClick` และ `List2_DbClick` เกือบจะเหมือนกัน ยกเว้นแต่จะถ่ายโอนชั้นรายการในทิศทางตรงกันข้าม



รูปที่ 4.11

แบบฟอร์มที่ออกแบบไว้สำหรับแอปพลิเคชัน Cheese

```
Sub Form_Load()
```

```
List1.AddItem "John"
List1.AddItem "Dang"
List1.AddItem "Manee"
List1.AddItem "Danny"
```

```

List1.AddItem "Jo"
List1.AddItem "David"
List1.AddItem "Linda"

End Sub

Sub List1_DblClick()

    List1.AddItem List1.Text

    List1.RemoveItem List1.ListIndex

End Sub

```

```

Sub List2_DblClick()

    List1.AddItem List2.Text

    List2.RemoveItem List2.ListIndex

End Sub

```

รูปที่ 4.12

รหัสโปรแกรมสำหรับแอปพลิเคชัน Cheese

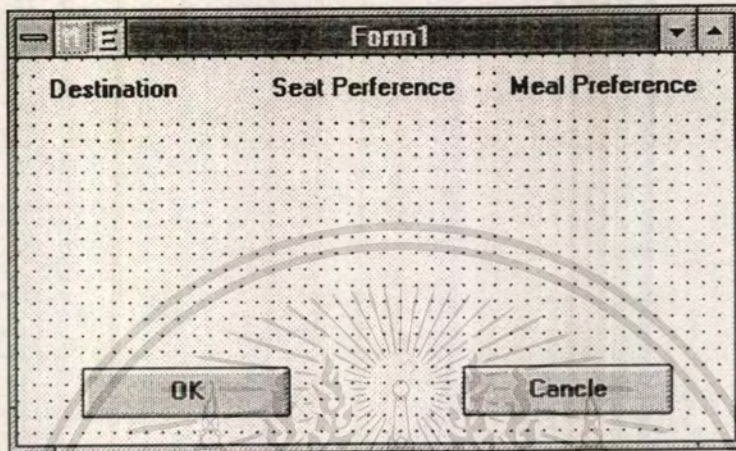
เมื่อคุณกดปุ่มเมาส์สองครั้งบนชั้นรายการหนึ่ง ๆ ที่อยู่ในลิสต์บ็อกซ์หนึ่ง ชั้นรายการนั้น จะถูกกำจัดออกจากบ็อกซ์นั้น และเติมเข้าไปในอีกบ็อกซ์หนึ่ง ลิสต์บ็อกซ์ List2 จะยังคงเรียงลำดับชั้นรายการที่ป้อนเข้าไปตามลำดับอักษรในขณะที่ List1 จะยังคงรักษาชั้นรายการที่ป้อนเข้าไปให้เรียงตามลำดับที่ได้เติมเข้าไป

การทดลองเกี่ยวกับ Combo Box

แอปพลิเคชัน Flight

โครงการนี้จะรวมเอาคอมโบบ็อกซ์ทั้งสามรูปแบบมาเรียงต่อกัน เพื่อที่ว่าคุณสามารถทดลองกับบ็อกซ์เหล่านั้น แอปพลิเคชันจะเลียนแบบไดอะล็อกซ์ที่สอบถามผู้ใช้เกี่ยวกับลักษณะความชอบสำหรับสายการบินหนึ่ง ๆ

1. สร้างโครงการใหม่ขึ้นหนึ่งโครงการ วางเลเบลสามเลเบลและปุ่มคำสั่งสองปุ่มบนแบบฟอร์มตั้งคุณสมบัติ Caption ของออบเจ็กต์แต่ละตัวให้ตรงกันกับข้อความจำหน้าตามที่แสดงไว้ในที่นี้



รูปที่ 4.13

2. ตั้งคุณสมบัติ Default ของปุ่ม OK ให้เป็น True และคุณสมบัติ Cancel ของปุ่ม Cancel ให้เป็น True
3. ได้เลเบล Destination ให้วาดคอมโบบ็อกซ์ และตั้งคุณสมบัติ Style ของบ็อกซ์นั้นให้เป็น 1 และตั้งคุณสมบัติ Name ของบ็อกซ์ให้เป็น cboDest ทำบ็อกซ์นี้ให้มีขนาดใหญ่พอที่จะแสดงข้อความออกมาสี่หรือห้าบรรทัดในส่วนของลิสต์บ็อกซ์
4. สร้างคอมโบบ็อกซ์อีกบ็อกซ์หนึ่งได้เลเบล Seat Preference ตั้งคุณสมบัติ Style ของบ็อกซ์ให้เป็น 2 และตั้งคุณสมบัติ Name ของบ็อกซ์ให้เป็น cboSeat
5. สร้างคอมโบบ็อกซ์เพิ่มเติมอีกหนึ่งบ็อกซ์ได้เลเบล Meal Preference ตั้งคุณสมบัติ Name ของบ็อกซ์ให้เป็น cboMeal และตั้งคุณสมบัติ Style ให้เป็น 0 แบบฟอร์มของคุณควรมีลักษณะเช่นที่แสดงในที่นี้

รูปที่ 4.14

6. ในที่นี้คุณจำเป็นต้องเขียนกระบวนการงานสองกระบวนการ โดยกระบวนการหนึ่งใช้รายงานผลลัพธ์ของการทำงานผู้ใช้เมื่อผู้ใช้กดปุ่มเมาส์บนปุ่มคำสั่ง OK และอีกกระบวนการหนึ่งเป็นการกำหนดค่าเริ่มต้นของคอมโบบ็อกซ์ กระบวนการส่วนหลังนี้จะเป็นส่วนหนึ่งของกระบวนการ *Form_Load* กระบวนการเหล่านี้ได้แสดงอยู่ในรูปที่ 4.15

```

Sub Command1_Click()
    Debug.Print cboDest.Text
    Debug.Print cboSeat.Text
    Debug.Print cboMeal.Text
End

End Sub

Sub Command2_Click()

End

End Sub

Sub Form_Load()
    cboDest.Text = ""
    cboDest.AddItem "Paris"
    cboDest.AddItem "Moscow"
    cboDest.AddItem "New York"

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

cboDest.AddItem "Cairo"
cboDest.AddItem "Melbourne"
cboDest.AddItem "Rio de Janeiro"
cboDest.AddItem "Tokyo"
cboSeat.AddItem "Aisle"
cboSeat.AddItem "Window"
cboSeat.AddItem "Center"

cboSeat.ListIndex = 1
cboMeal.AddItem "Chicken"
cboMeal.AddItem "Lasagne"
cboMeal.AddItem "Vageterian"
cboMeal.AddItem "Kosher"
cboMeal.AddItem "No preference"

End Sub

```

รูปที่ 4.15

รหัสโปรแกรมสำหรับแอปพลิเคชัน Flight

เมื่อคุณเอ็กซ์คิวด์แอปพลิเคชัน ลองดูว่าคอมโบบ็อกซ์ใดที่ให้คุณสามารถพิมพ์ข้อมูลเข้าไป ลักษณะปรากฏของคอมโบบ็อกซ์แต่ละชนิดตามที่แสดงอยู่ในรูปที่ 4.16 ยังช่วยบอกเป็นนัยให้ทราบว่า จะดำเนินการขั้นตอนต่อไปอย่างไร

รูปที่ 4.16

การดำเนินงานแอปพลิเคชัน Flight

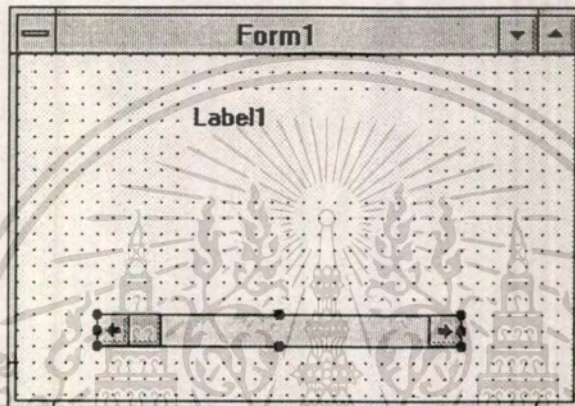
เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การทดลองเกี่ยวกับ Scroll Bar

แอปพลิเคชัน Scroll Bar

โปรแกรมนี้จะเพียงแค่แสดงให้เห็นผลป้อนกลับเกี่ยวกับแนวทางการทำงานของสกรอลล์บาร์

1. สร้างโครงการใหม่ขึ้นหนึ่งโครงการบนแบบฟอร์มว่าง ๆ นี้ให้วางฟิลด์เลเบลและสกรอลล์บาร์ตามแนวนอนเช่นที่แสดงในที่นี้



รูปที่ 4.17

2. ตั้งคุณสมบัติ BorderStyle ของเลเบล 1-Fixed Single และตั้งคุณสมบัติ Caption ให้เป็นสตริงว่าง ตั้งคุณสมบัติของสกรอลล์บาร์ตามแนวนอนให้เป็นอย่างต่อไปนี้

คุณสมบัติ	ค่ากำหนด
LargeChange	10
Max	100
Min	0
SmallChange	2

3. กดปุ่มเมาส์สองครั้งบนสกรอลล์บาร์เพื่อก่อให้เกิดวินโดว์ห้สขึ้น และป้อนกระบวนการงานนี้เพื่อจัดการกับอีเวนต์ Change

```
Sub HScroll1_Change()
    Label1.Caption = Str$(HScroll1.Value)
End Sub
```

เนื่องจากคุณสมบัติ Caption ของเลเบลจะรับเฉพาะค่าสตริงเท่านั้น ดังนั้นคุณจึงต้องใช้ฟังก์ชัน Str\$ เพื่อแปลงค่าของ HScroll1.Value ให้ไปเป็นสตริง

4. ป้อนกระบวนการงานต่อไปนี่เพื่อจัดการกับอีเวนต์ Scroll

```
Sub HScroll1_Scroll()
```

```
Label1.Caption = "Moving to" + Str$(HScroll1.Value)
```

```
End Sub
```

ในตอนนี้เริ่มใช้งานแอปพลิเคชัน เมื่อคุณคลิกปุ่มเมาส์เข้าไปในบริเวณสี่เหลี่ยมของสกรอลล์บาร์ค่าที่แสดงอยู่ในเลเบลจะเปลี่ยนแปลงในหน่วยของ 10 เมื่อคุณคลิกปุ่มเมาส์บนลูกศรเลื่อนจอภาพไม่ว่าจะเป็นปลายด้านใดของสกรอลล์บาร์ ค่าจะเปลี่ยนแปลงในหน่วยของ 2 เมื่อลากสกรอลล์บ็อกซ์ด้วยเมาส์ คุณสามารถปรับค่าในหน่วยที่นอกเหนือจาก 2 หรือ 10 เช่นค่าที่แสดงอยู่ในรูปที่ 4.18 อาจทำให้เกิดขึ้นโดยการคลิกปุ่มเมาส์บนลูกศรเลื่อนจอภาพ หรือโดยการลากสกรอลล์บ็อกซ์ ลองสังเกตดูจะเห็นว่ากระบวนการสำหรับอีเวนต์ Scroll ปรับข้อความจำหน้าของเลเบลในขณะที่คุณลากสกรอลล์บ็อกซ์ เลือก Break จากเมนู Run หรือคลิกปุ่มเมาส์บนปุ่ม Break ของทูลบาร์ และหลังจากนั้นป้อนข้อความสั่ง HScroll1.Value = 17 ในวินโดว์ Debug จะเห็นว่าตำแหน่งของสกรอลล์บ็อกซ์จะเปลี่ยนแปลงอีกครั้ง โดยจะยังคงรักษาค่าตำแหน่งที่เหมาะสมเทียบกับค่ากำหนด Min และ Max



รูปที่ 4.18

การดำเนินงานแอปพลิเคชัน Scroll Bar

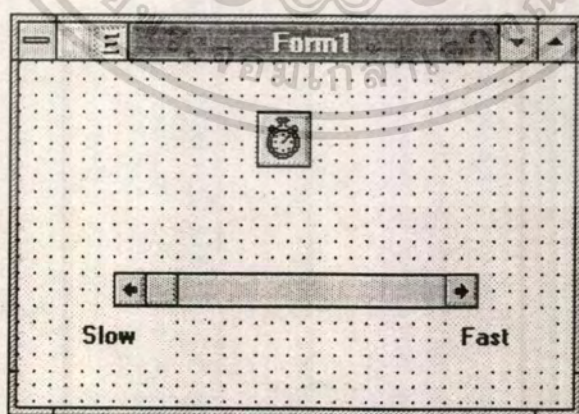
การทดลองเกี่ยวกับ Timer

แอปพลิเคชัน Metronome

โครงการนี้ ซึ่งสร้างเครื่องทำจังหวะหรือเมโทรโนม โดยการใช้ตัวจับเวลาหรือสกรอลล์บาร์จะแสดงให้เห็นว่ามีอะไรเกิดขึ้น เมื่อคุณพยายามใช้ออบเจกต์ตัวจับเวลาเป็นเครื่องจับเวลาที่มีความแม่นยำอย่างสูง ตามปกติเครื่องทำจังหวะจะดำเนินการอยู่ในพิสัยตั้งแต่ประมาณ 40 จังหวะต่อนาทีสำหรับเพลงช้า ๆ ไปจนถึง 200 จังหวะต่อนาทีสำหรับเพลงเร็วมาก ๆ การตั้งให้ 40 จังหวะต่อนาทีที่เทียบเท่ากับ $\frac{2}{3}$ จังหวะต่อวินาที หรือหมายถึง 1 จังหวะทุก ๆ 1500 มิลลิวินาที ตรงอีกปลายด้านหนึ่งของสเกล 200 จังหวะต่อนาทีที่เทียบเท่ากับ 3 เศษ $\frac{1}{3}$ จังหวะต่อวินาทีหรือ 1 บิตทุก ๆ 300 มิลลิวินาที ดังนั้นจึงอาจใช้ค่า 300 และ 1500 เป็นจุดสูงสุดและจุดต่ำสุดสำหรับสกรอลล์บาร์ในแอปพลิเคชัน (ค่ายิ่งสูง บิตจะยิ่งต่ำ ฉะนั้น 1500 เป็นค่าต่ำสุด ไม่ใช่ค่าสูงสุด)

1. สร้างโครงการใหม่ขึ้นหนึ่งโครงการ และเพิ่มเติมสกรอลล์บาร์ตามแนวนอนเข้าไปในแบบฟอร์ม หลังจากนั้นสร้างออบเจกต์ตัวจับเวลาด้วยการกดปุ่มเมาส์สองครั้งบนเครื่องมือสร้างตัวจับเวลาที่อยู่ในทูลบ็อกซ์ คุณไม่จำเป็นต้องวาดหรือเปลี่ยนขนาดออบเจกต์ตัวจับเวลา ทั้งนี้เพราะจะแสดงออบเจกต์นี้ออกมาบนแบบฟอร์มในเวลาออกแบบ แต่จะไม่แสดงออกมาเมื่อโปรแกรมกำลังดำเนินงาน

2. วางเลเบลไว้ได้ปลายด้านซ้ายของสกรอลล์บาร์ และตั้งคุณสมบัติ Caption ของเลเบลให้เป็น Slow ได้ปลายด้านขวาของการสกรอลล์บาร์ ให้สร้างเลเบลพร้อมกับมีข้อความจำเป็นเป็น Fast สิ่งที่คุณควรจะเห็นบนจอภาพเป็นดังนี้



รูปที่ 4.19

3. ตั้งคุณสมบัติ Interval ของออบเจกต์ตัวจับเวลาให้เป็น 1000

เอกสารนี้เป็นเอกสารทบทวนใจสำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า

ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. ตั้งคุณสมบัติสกรอลล์บาร์ตามแนวนอนให้เป็นดังต่อไปนี้

คุณสมบัติ	ค่ากำหนด
LargeChange	100
Max	300
Min	1500
SmallChange	25
Value	1000

5. คุณจำเป็นต้องเติมรหัสโปรแกรมสองบรรทัดนี้เข้าไปในแอปพลิเคชันนี้ โดยบรรทัดหนึ่งป้อนเข้าไปในกระบวนการงาน Change ของสกรอลล์บาร์ และอีกบรรทัดหนึ่งในกระบวนการงาน Timer ของออบเจกต์ตัวจับเวลา กระบวนการงาน Change จะเก็บออบเจกต์ตัวจับเวลาให้เกี่ยวข้องกับสกรอลล์บาร์ และกระบวนการงาน Timer จะก่อกำเนิดเสียงดังกของเครื่องทำจิ้งหะ (หรือในกรณีคือเสียงบี๊ป)

```
Sub HScroll1_Change( )
    Timer1.Interval = HScroll1.Value
End Sub
Sub Timer1_Timer( )
    Beep
End Sub
```

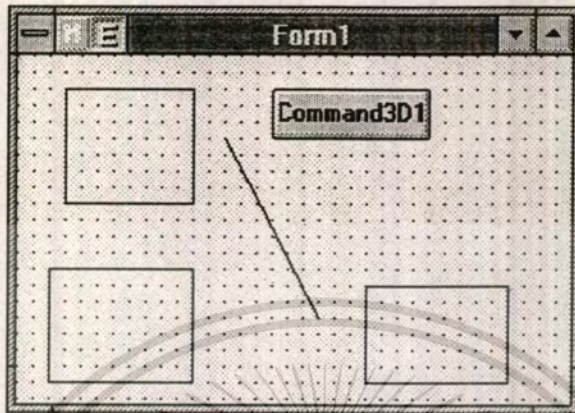
เมื่อคุณเริ่มใช้งานแอปพลิเคชัน เครื่องทำจิ้งหะจะเกิดเสียงที่ 1 จิ้งหะต่อวินาที การเลื่อนสกรอลล์บ็อกซ์จะเปลี่ยนแปลงความเร็วของการเกิดเสียงดังก ลองพยายามโยกย้ายวินโดว์ไปรอบ ๆ บนจะภาพหรือดำเนินงานแอปพลิเคชันอีกโปรแกรมหนึ่งในส่วนหลัง (background) เมื่อไม่มีสิ่งใด ๆ กำลังดำเนินอยู่นั้น ตัวจับเวลาค่อนข้างแม่นยำมาก แต่คุณจะได้เห็นว่าอีเวนต์ของตัวจับเวลาบางอีเวนต์สูญเสียลำดับไปอย่างไรเมื่อคอมพิวเตอร์สับเปลี่ยนความสนใจไปยังการทำงานอื่น ๆ

การทดลองเกี่ยวกับ Shape , Image and Line

โครงการ Layer

โครงการง่าย ๆ นี้ไม่ได้เป็นแอปพลิเคชันสมบูรณ์แบบ แต่จะให้คุณได้ทดลองกับออบเจกต์แสดงผล เพื่อที่ว่า คุณจะได้อุ่นเคยกับเครื่องมือมากยิ่งขึ้น เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

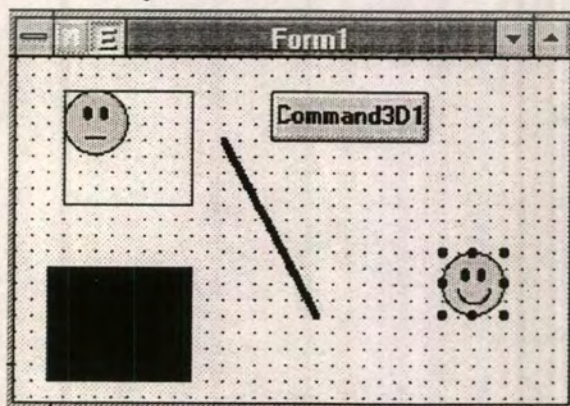
1. สร้างโครงการใหม่ขึ้นโครงการบนแบบฟอร์ม ให้วางบ็อกซ์ภาพ ปุ่มคำสั่ง รูปทรงเส้น ตัวควบคุมภาพลักษณ์ ดังที่แสดงในที่นี้



รูปที่ 4.20

2. เลือกคุณสมบัติ Picture ของ Picture1 ในวินโดว์ Properties กดปุ่มเมาส์บนปุ่มไขปลาที่อยู่ด้านข้างของเซตติงบ็อกซ์เพื่อแสดงไดอะล็อกบ็อกซ์ Load Picture ออกมา ในไดอะล็อกบ็อกซ์ให้ตั้งชนิดของไฟล์เพื่อแสดงชื่อไฟล์ที่มีส่วนขยาย ICO ในไดเรกทอรีย่อย MISC ของไดเรกทอรี ICONS ที่มาพร้อมกับ Visual Basic คุณจะพบไฟล์ไอคอน FACE01.ICO เลือกไฟล์นี้เพื่อบรรจุไอคอนเข้าไปในบ็อกซ์ภาพ ในรูปแบบทำนองเดียวกันนี้ ให้ตั้งคุณสมบัติ Picture ของ Image1 เพื่อบรรจุไอคอน FACE02.ICO ในตัวควบคุมภาพลักษณ์

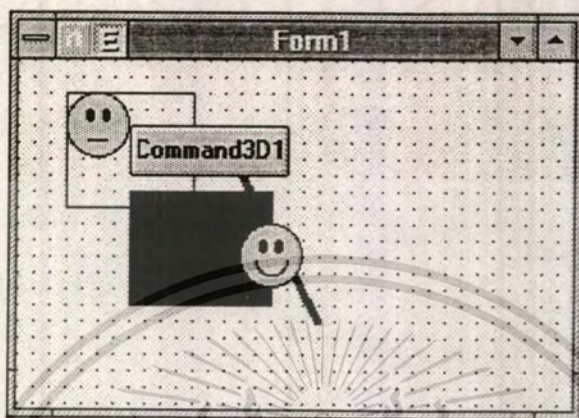
3. ตั้งคุณสมบัติ FillStyle ของ Shape1 ให้เป็น 0 - Solid และคุณสมบัติ FillColor ให้เป็น red (กดปุ่มเมาส์บนปุ่มไขปลาที่อยู่ด้านข้างเซตติงบ็อกซ์ และเลือก red จากชุดอุปกรณ์สี) ตั้ง BorderWidth ของ Line1 ให้เป็น 4 รูปแบบควรจะเป็นเช่นดังนี้



รูปที่ 4.21

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

4. โยกย้ายรูปทรงให้ไปอยู่ที่ตรงกลางของแบบฟอร์ม และหลังจากนั้นโยกย้ายออบเจกต์อื่น ๆ เอาไว้เหนือรูปทรง เพื่อที่จะทำให้เลขบางส่วนเช่นที่แสดงในที่นี้



รูปที่ 4.22

ขอให้สังเกตว่าในแต่ละชั้นของการแสดงผล ออบเจกต์ที่สร้างขึ้นล่าสุดจะปรากฏทับอยู่บนออบเจกต์ที่สร้างขึ้นก่อนหน้านี้ (บ็อกซ์ภาพและคำสั่งวางตัวอยู่บนชั้นบนสุด รูปทรง เส้นและภาพลักษณะ จะอยู่ตรงชั้นกลาง)

ลำดับซึ่งออบเจกต์บางชนิดซ้อนออบเจกต์อีกตัวหนึ่งนั้นเรียกว่า ลำดับ Z และสามารถเปลี่ยนแปลงได้ ให้กดปุ่มเมาส์บน Line1 เพื่อเลือกออบเจกต์นั้น และเลือกคำสั่ง Bring To Front จากเมนู Edit ของ Visual Basic เส้นซึ่งปรากฏอยู่เหนือรูปทรงแล้วนั้น ในตอนนี้จะปรากฏอยู่เหนือภาพลักษณะด้วย อย่างไรก็ตาม เมื่อคุณเคลื่อนย้ายเส้นไปรอบ ๆ จอภาพ เส้นจะไม่ปกคลุมบ็อกซ์ภาพหรือคำสั่ง คำสั่ง Bring To Front (และคำสั่ง Send To Back) จะดำเนินการเฉพาะภาพในชั้นใดชั้นหนึ่งเท่านั้น ออบเจกต์ชั้นกลางจะปรากฏอยู่หลังออบเจกต์ชั้นบนเสมอ คุณสามารถใช้คำสั่ง Bring To Front และ Send To Back กับตัวควบคุมทั้งหมด ซึ่งทำให้คุณสามารถปรับลำดับ Z ภายในชั้นแต่ละชั้น ลองพยายามใช้ Send To Back หลังจากเลือกปุ่มคำสั่งแล้ว

การทดลองเกี่ยวกับ File

การสร้างแอปพลิเคชัน TestFile

เราใช้แอปพลิเคชัน TestFile แสดงให้เห็นว่าจะสร้างโคเดิล็อกบ็อกซ์โดยใช้แบบฟอร์มของ Visual Basic และเครื่องมือสร้างระบบไฟล์สามอย่างขึ้นได้อย่างไร รหัสและเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ได้อะลือกบ็อกซ์ของ TestFile สามารถนำกลับมาใช้ได้ อีก ดังนั้นจึงอาจคัดลอกเข้าไปในแอปพลิเคชันอื่น ๆ ที่ต้องการความสามารถอย่างเดียวกัน

เริ่มต้นด้วยการเปิดโครงการใหม่ขึ้นหนึ่งโครงการ ตั้งคุณสมบัติ Caption ของแบบฟอร์มให้เป็น Main และตั้งคุณสมบัติ Name ของแบบฟอร์มให้เป็น frmMain วางปุ่มคำสั่งเอาไว้ตรงกลางของแบบฟอร์ม และตั้งคุณสมบัติ Name ของปุ่มให้เป็น btnTest และคุณสมบัติ Caption ของปุ่มให้เป็น Test กดปุ่มเมาส์สองครั้งบนปุ่มคำสั่งเพื่อเปิดวินโดว์รหัสและนิยามกระบวนการ Click ของปุ่มให้เป็นตามที่แสดงในที่นี้

```
Sub btnTest__Click( )
```

```
Dim Selected As String
```

```
Selected = OpenFileDialog ( "*.TXT" )
```

```
If Selected = "" Then
```

```
MsgBox "Canceled"
```

```
Else
```

```
MsgBox "File " + Selected + " selected"
```

```
End If
```

```
End Sub
```

กระบวนการนี้เรียกฟังก์ชัน OpenFileDialog (ยังไม่ได้นิยาม) ซึ่งแสดงได้อะลือกบ็อกซ์ที่อนุญาตให้ผู้เลือกใช้ไฟล์โดยอาศัยข้อกำหนดคุณลักษณะไฟล์ *.TXT ถ้าผู้ใช้ไม่ได้เลือกไฟล์ และเพียงแต่เปิดได้อะลือกบ็อกซ์ ฟังก์ชันจะส่งคืนสตริงว่าง มิฉะนั้นแล้วกระบวนการ btnTest__Click จะแสดงชื่อของไฟล์ที่เลือกไว้ออกมา

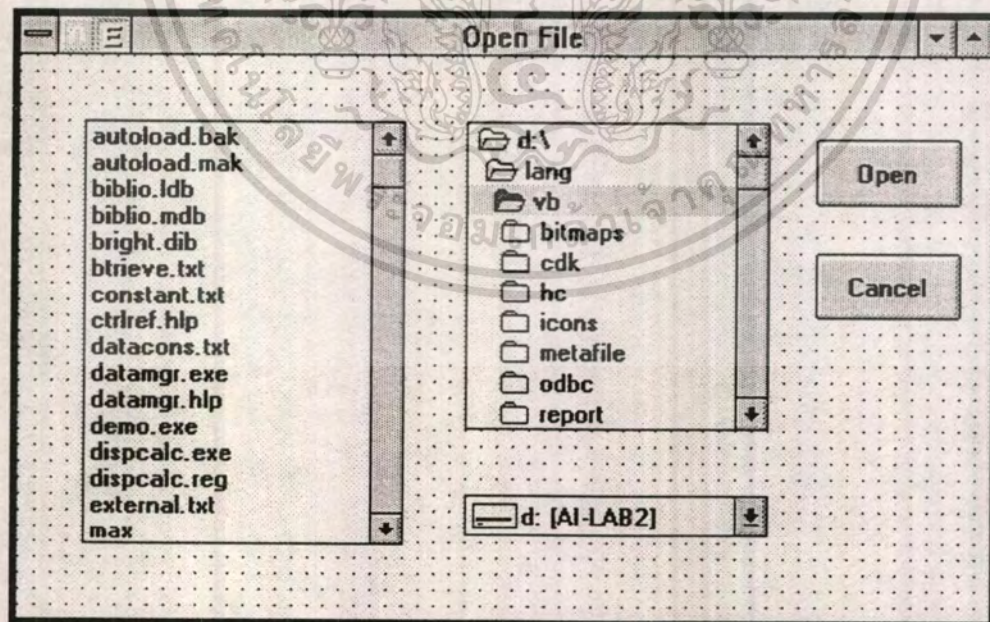
การเติมแบบฟอร์มตัวที่สอง

ขั้นตอนถัดไปเป็นการสร้างได้อะลือกบ็อกซ์ที่ฟังก์ชัน OpenFileDialog จะแสดงออกมา เลือกคำสั่ง New Form จากเมนู File หรือกดปุ่มเมาส์บนปุ่ม New Form บนทูลบาร์เพื่อระบุแบบฟอร์มตัวที่สองของ Visual Basic เอาไว้บนจอภาพ บนแบบฟอร์มใหม่นี้ให้เติมไฟล์ลิสต์บ็อกซ์ ใดเรกทอรีลิสต์บ็อกซ์ ใดฟิลิสต์บ็อกซ์ และปุ่มคำสั่งสองปุ่ม ตั้งคุณสมบัติของออบเจกต์เหล่านี้ตามที่แสดงในตารางที่ 4.2 และหลังจากนั้นจัดออบเจกต์บนแบบฟอร์มให้คล้ายคลึงกับรูปที่ 4.23

ออบเจ็กต์	คุณสมบัติ	ค่ากำหนด
Form2	BorderStyle	3 - Fixed Double
	Caption	Open File
	Name	frmOpen
File1	Name	fileList
Dir1	Name	dirList
Drive1	Name	drvList
Button1	Caption	Open
	Default	True
	Name	btnOpen
	Cancle	True
Botton2	Caption	Cancle
	Name	btnCancle

ตารางที่ 4.2

การตั้งคุณสมบัติสำหรับแบบฟอร์มตัวที่สองในแอปพลิเคชัน TestFile

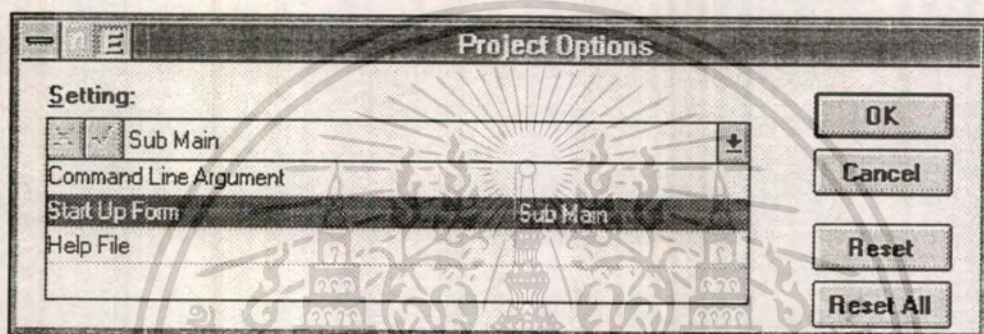


รูปที่ 4.23

การออกแบบฟอร์มสำหรับไดอะล็อกบ็อกซ์ Open File ในแอปพลิเคชัน TestFile

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ในเวลาที่สร้างแอปพลิเคชันของ Visual Basic ที่มีมากกว่าหนึ่งแบบฟอร์ม Visual Basic จะโหลดแบบฟอร์มเพียงหนึ่งแบบในเวลาที่เริ่มต้นใช้งาน ในการที่จะระบุให้ frmMain เป็นแบบฟอร์มเริ่มใช้งาน ให้เลือก ProJect จากเมนู Option ในไดอะล็อกบ็อกซ์ Priject Option ตามที่แสดงอยู่ในรูปที่ 4.24 เลือกขึ้นรายการ Start Up Form ตรงบริเวณส่วนบนของไดอะล็อกบ็อกซ์มี Setting drop-down list box ที่ประกอบไปด้วยรายการของแบบฟอร์มในแอปพลิเคชัน เลือก frmMain และกดปุ่มเมาส์บนปุ่ม OK



รูปที่ 2.24

ไดอะล็อกบ็อกซ์ Project Options

หมายเหตุ ในเวลาที่เลือกขึ้น Start Up Form ที่อยู่ในไดอะล็อกบ็อกซ์ Project Options สังเกตดูจะเห็นว่าเว็คดิงลิสต์บ็อกซ์ประกอบไปด้วยข้อขึ้น Sub Main รวมทั้งรายการของแบบฟอร์มที่อยู่ในแอปพลิเคชัน การเลือกข้อขึ้น Sub Main จะบังคับ Visual Basic ไม่ให้โหลดแบบฟอร์มในเวลาที่โปรแกรมเริ่มต้นโดยการดำเนินงานกระบวนการงานย่อยชื่อว่า Main เป็นการแทนที่ การต้องใช้ข้อขึ้นนี้อาจเป็นในกรณีเช่นแบบฟอร์มที่โปรแกรมเปิดขึ้นอยู่กับวันของสัปดาห์ หรือขึ้นกับพารามิเตอร์กำหนดค่าเริ่มต้นบางค่า

การเติมมอดูลรหัส (Module)

สิ่งถัดไปที่จะต้องทำคือ การเขียนกระบวนการที่ควบคุมไดอะล็อกบ็อกซ์บ็อกซ์ Open File รวมทั้งฟังก์ชัน OpenFileDialog คุณอาจจะคาดหมายให้รหัสนี้ทั้งหมดอยู่ในแบบฟอร์ม frmOpen แต่มีข้อควรคำนึงที่สำคัญประการหนึ่งที่ทำให้ไม่สามารถทำเช่นนั้น นั่นคือขอบเขตของเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

กระบวนการทั้งหมดในมอดูลของแบบฟอร์มจะถูกจำกัดให้อยู่กับมอดูลนั้น กล่าวคือกระบวนการใด ๆ ใน frmOpen จะสามารถเข้าถึงได้เฉพาะจากกระบวนการอื่น ๆ ใน frmOpen เนื่องจากการเรียกไปยัง OpenFileDialog อยู่ใน frmMain ดังนั้นจึงไม่สามารถวางฟังก์ชันเอาไว้ใน frmOpen

ทางแก้ทางหนึ่งอาจจะทำได้คือ บรรจุรหัสสำหรับ OpenFileDialog เอาไว้ใน frmMain แต่เราสนใจสร้างโค้อบล็อกที่สามารถใช้ซ้ำได้ และปุ่ม Test ใน frmMain ไม่จำเป็นที่จะต้องใช้ในแอปพลิเคชันทุกโปรแกรมดังนั้นทางแก้ที่ดีกว่าคือ การสร้างมอดูลรหัสแยกออกไป มอดูลรหัสนี้ประกอบไปด้วยรหัสที่ควบคุมโค้อบล็อก เนื่องจากกระบวนการทุก ๆ กระบวนการมีมอดูลรหัสมีขอบเขตแบบครอบคลุม ฉะนั้นจึงสามารถแก้ปัญหาเกี่ยวกับการเข้าถึงได้ มอดูลรหัสมีลักษณะเป็นแอนติตี้ที่แยกต่างหาก ซึ่งสามารถจะโหลดเข้าไปไว้ในแอปพลิเคชันอื่น ๆ ของ Visual Basic

เติมมอดูลรหัสโดยการเลือก New Module จากเมนู File ของ Visual Basic หรือโดยการกดปุ่มเมาส์บนปุ่ม New Module ของทูลบาร์ Visual Basic จะเติมไฟล์ชื่อว่า MODULE1.BAS เข้าไปในรายการวินโดว์ Project และจะเปิดวินโดว์ของรหัสมอดูลขึ้น ในส่วนการประกาศทั่วไปของมอดูลให้เติมข้อความดังต่อไปนี้

Global SelectedFile As String

ข้อความสั่งนี้ทำหน้าที่ประกาศตัวแปรสตริงครอบคลุมที่จะครอบครองชื่อไฟล์ที่ผู้ใช้เลือกตัวแปรจะต้องเป็นแบบครอบคลุม เพราะจะต้องเข้าถึงได้จากฟังก์ชัน OpenFileDialog ในมอดูลรหัส และจากกระบวนการอื่นใน frmOpen

ในตอนนี้เติมกระบวนการต่อไปนี้เข้าไปในมอดูลรหัส

Function OpenFileDialog (Wild As String) As String

Load frmOpen

frmOpen.fileList.Pattern = Wild

SelectedFile

frmOpen.Show 1

OpenFileDialog = SelectedFile

End Function

ฟังก์ชันนี้จะถูกเรียกพร้อมกับมีข้อกำหนดคุณลักษณะไฟล์วิลด์การ์ด (เช่น *.TXT) เป็นอาร์กิวเมนต์ ในขั้นตอนแรกฟังก์ชันจะโหลดแบบฟอร์ม frmOpen เข้าไปในหน่วยความจำโดยให้ข้อความสั่ง Load ข้อความสั่ง Load ทำให้อย่างมั่นใจว่ามีแบบฟอร์มนี้อยู่ แต่จะไม่ทำให้แบบฟอร์มทำให้สามารถเห็นได้ รหัสบรรทัดถัดไปตั้งคุณสมบัติ Pattern ของออบเจกต์ fileList ให้แก่แบบแผนเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

วิธการ คุณสมบัติ Pattern ของไฟล์ลิสต์บ็อกซ์จะกำหนดว่ามีไฟล์ใดบ้างที่ปรากฏอยู่ในรายการ เช่นแบบแผน *.* จะทำให้แสดงไฟล์ทั้งหมดออกมา ในฟังก์ชันนี้จะเขียนรหัสในลักษณะ form.object.property แทนที่จะเป็นการระบุ object.property แบบทั่ว ๆ ไป ด้วยเหตุที่ฟังก์ชัน OpenFileDialog ไม่ได้อยู่ในแบบฟอร์ม frmOpen ถัดไปจะดัดแปลง SelectedFile ให้เป็นสตริงว่าง ซึ่งแสดงว่ายังไม่ได้มีการเลือกใด ๆ ทั้งสิ้น ข้อความที่ตามมาเรียกว่ากลวิธี Show สำหรับ frmOpen ซึ่งทำแบบฟอร์มให้อยู่ในสภาพที่เป็นได้ Show อาจเรียกโดยมีอาร์กิวเมนต์เป็น 0 หรือ 1 เมื่อใช้ Show 0 แบบฟอร์มจะเปลี่ยนมาอยู่ในสภาพที่เห็นได้ แต่การประมวลยังคงดำเนินต่อไปในกระบวนการที่เรียกกลวิธี Show เมื่อใช้ Show 1 ดังเช่นในกรณีนี้ แบบฟอร์มที่ระบุไม่เพียงแต่จะเปลี่ยนเป็นสภาพที่เห็นได้ แต่ยังสามารแสดงออกมาตามแบบนั้นคือไม่สามารถใช้แบบฟอร์มอื่น ๆ ในแอปพลิเคชันในขณะที่แสดงแบบฟอร์มนี้ออกมาทางจอภาพ การประมวลจะหยุดชั่วคราวในกระบวนการเรียกจนกว่าจะได้ปิดแบบฟอร์มตามแบบ

เมื่อผู้ใช้ปิดไดอะล็อกบ็อกซ์ซึ่งซ่อน frmOpen อย่างมีประสิทธิภาพ ฟังก์ชัน OpenFileDialog การลกรหัสกระบวนการอื่น

สิ่งที่เหลืออยู่คือ การลกรหัสกระบวนการอื่นสำหรับออบเจ็กต์ใน frmOpen เรานิยามกระบวนการเหล่านั้นไว้ในรูปที่ 4.25 หมายเหตุที่อยู่ในรหัสจะบรรยายว่ามีอะไรเกิดขึ้นในกรณีที่โปรแกรมเอ็กซีคิวต์ เมื่อพิจารณาในโปรแกรมจะสามารเห็นได้ว่าการดำเนินการของออบเจ็กต์ระบบไฟล์ค่อนข้างง่ายแม้ว่าจะต้องเขียนโปรแกรมการโต้ตอบกันในหมู่ของออบเจ็กต์อย่างระมัดระวังก็ตาม

```
Sub filList_DblClick( )
```

```
    'Double-Clicking on a filename is the same as
```

```
    'clicking on the Open button
```

```
    btnOpen_Click
```

```
End Sub
```

```
SubdirList_Change( )
```

```
    "When the directory list box change, set a
```

```
    'new path for the list box
```

```
    filList.Path = dirList.Path
```

```
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Sub drvList_Change( )
    'When the drive list box changes, set a
    'new path for the directory list box
    'Also causes a Change event for dirList
    dirList.Path = drvList.Drive
End Sub

Sub btnCancel_Click( )
    'No file selected; clear the string and Hide the form
    SelectedFile = ""
    Hide
End Sub

Sub btnOpen_Click( )
    'Be sure a filename has been highlighted in the
    'file list box
    If filList.ListIndex >= 0 Then
        'Get the path of the selected file
        SelectedFile = filList.Path
        'Be sure the Pathname ends with a backslash
        If Right$(SelectedFile, 1) <> "\" Then
            SelectedFile = SelectedFile + "\"
        End If
        'Append the selected filename to the path
        SelectedFile = SelectedFile + filList.List(filList.ListIndex)
    End If
    'Hide the form (close the dialog box)
    Hide
End Sub

```

รูปที่ 4.25

รหัสโปรแกรมสำหรับกระบวนการงานอีเวนต์ในแอปพลิเคชัน TestFile

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

การเอ็กซ์คิวิต์และการเก็บแอปพลิเคชัน TestFile

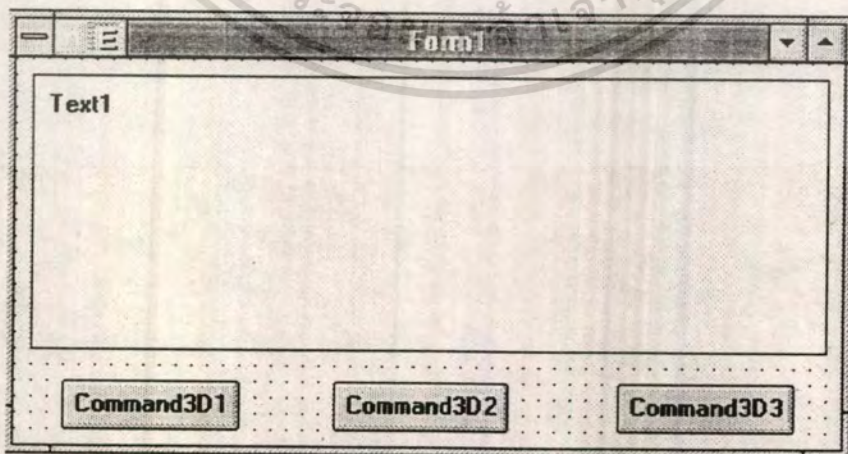
ในเวลาที่เริ่มใช้งานแอปพลิเคชัน TestFile เราจะเห็นเฉพาะแบบฟอร์มเริ่มต้นที่กำหนดไว้ ซึ่งจะบรรจุปุ่ม Test กดปุ่มเมาส์บนปุ่ม Test เพื่อแสดงไดอะล็อกบ็อกซ์ Open File เลือกไดเรกทอรีและไฟล์ที่ต่างออกไปเพื่อดูว่าตัวควบคุมมีปฏิกิริยาโต้ตอบเป็นอย่างไรบ้างเมื่อเลือกไฟล์ เท็กซ์บ็อกซ์จะแจ้งให้ทราบเกี่ยวกับตัวเลือกที่เลือกไว้ ถ้าไม่ได้เลือกไฟล์ หรือถ้ากดปุ่มเมาส์บนปุ่ม Cancel เท็กซ์บ็อกซ์จะชี้ให้เห็นว่าได้ยกเลิกไดอะล็อกบ็อกซ์ เราสามารถกดปุ่มเมาส์บนปุ่ม Test และทำให้เกิดไดอะล็อกบ็อกซ์ขึ้นได้มากครั้งตามความต้องการ

คุณจะได้เห็นได้ว่าการสร้างความพยายามของ Common Dialog Control ขึ้นมาใหม่จะต้องอาศัยความพยายามค่อนข้างมาก เราสามารถสร้างแอปพลิเคชัน TestFile โดยใช้แบบฟอร์มเดียว ๆ แบบฟอร์มเดียว และมีรหัสประมาณ 10 บรรทัดในกระบวนการ `blnTest_Click` ถ้าคุณใช้ Common Dialog Control

ขอให้นึกได้ว่าได้เก็บแอปพลิเคชัน TestFile ไว้เพราะเราสามารถใช้ฟังก์ชัน `OpenFileDlg` ในโครงการอื่น ๆ ได้อีก เมื่อจะบูตแอปพลิเคชันให้เลือกคำสั่ง Save Project จากเมนู File ใช้ชื่อไฟล์ `FRMNAIN.FRM, FRMOPEN.FRM` และ `FRMOPEN.BAS` สำหรับแบบฟอร์มหลัก แบบฟอร์มไดอะล็อกบ็อกซ์และมอดูลที่สเก็ปไฟล์โครงการเป็น `TESTFILE.MAK` การทดลองเกี่ยวกับ Sequential File

สร้างโครงการใหม่ขึ้นหนึ่งโครงการ และวางเท็กซ์บ็อกซ์ขนาดใหญ่หนึ่งบ็อกซ์ และปุ่มคำสั่งสามปุ่มบนแบบฟอร์มตามที่แสดงในรูปที่ 4.26 ตั้งคุณสมบัติของออบเจกต์ตามที่ลงในตารางที่

4.3



รูปที่ 4.26

การออกแบบแบบฟอร์มตอนแรกเริ่มสำหรับแอปพลิเคชัน Configurator

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ออบเจกต์	คุณสมบัติ	การกำหนดค่า
Form1	BorderStyle	3 - Fixed Double
	Caption	Configurator
	Name	Cedit
Text1	MultiLine	True
	Name	txtContent
	ScrollBars	3 - Both
	Text	[none]
Command1	Caption	Open
	Name	btnOpen
Command2	Caption	Save
	Name	btnSave
Command3	Caption	Quit
	Name	btnQuit

ตารางที่ 4.3

การตั้งคุณสมบัติสำหรับแอปพลิเคชัน Configurator

ในตอนการประกาศทั่วไปของรหัส ให้เติมการประกาศดังต่อไปนี้

Dim TextChanged As Integer, ConfigFound As Integer, CRLF As String

หลังจากนั้น สำหรับออบเจกต์ที่เหมาะสม ป้อนกระบวนการตามที่แสดงในรูปที่ 4.27

```
Sub Form_Load ( )
```

```
    TextChanged = False
```

```
    ConfigFound = False
```

```
    CRLF = Chr$(13) + Chr$(10)
```

```
End Sub
```

```
Sub btnOpen_Chick( )
```

```
    Dim Cline As String
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

if ConfigFound Then
    Beep
    Exit Sub

End if

Open "C:\CONFIG.SYS" For Input As # 1
Do While Not EOF(1)
    Line Input # 1, Cline
    txtContent.Text = txtContent.Text + Cline + CRLF
Loop
close # 1
ConfigFound = True
End Sub

Sub btnSave_Click( )
    if Not (ConfigFound And TextChanged) Then
        Beep
        Exit Sub
    End if
    Open "C:\CONFIG.SYS" For Output As # 1
    Print # 1, txtContent.Text
    Close # 1
End Sub

Sub btnQuit_Click( )
    End
End Sub

Sub txtContent_Change( )
    TextChanged = True
End Sub

```

รูปที่ 4.27

รหัสโปรแกรมสำหรับแอปพลิเคชัน Configurator

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

สังเกตดูในรหัสโปรแกรม Configurator จะมีตัวแปรสตริง CRLF ปรากฏอยู่ ตามปกติจะชอบการทำให้เกิดการสิ้นสุดบรรทัดเอาไว้ ข้อความสั่ง Line Input # จะอ่านไปจนถึงจุดสิ้นสุดของบรรทัด และหลังจากนั้นผ่านสตริงที่ประกอบไปด้วยอักขระทั้งหมดในบรรทัด ยกเว้นเครื่องหมายยุติบรรทัด (line terminator) ข้อความสั่ง Print # จะเขียนในสิ่งที่ระบุไว้ และผนวกอักขระสิ้นสุดบรรทัด (end-of-line) เข้าไว้ด้วยโดยอัตโนมัติ อย่างไรก็ตาม ในโปรแกรมตัวอย่างได้แทรกอักขระปิดแคร์และอักขระป้อนบรรทัดที่ได้ถูกตัดออกไปด้วยข้อความสั่ง Line Input # เข้าไปใหม่เพื่อที่เท็กซ์บ็อกซ์จะได้ทราบว่าบรรทัดของบรรทัดอยู่ตรงที่ใด และสามารถจัดรูปแบบจอแสดงผลได้อย่างเหมาะสม



รูปที่ 4.28

การแก้ไขไฟล์ CONFIG.SYS ด้วยแอปพลิเคชัน Configurator

รูปที่ 4.28 แสดงให้เห็นว่าแอปพลิเคชัน Configurator มีลักษณะอย่างไรในเวลาที่ดำเนินงานโปรแกรม และใช้โปรแกรมแก้ไขไฟล์ CONFIG.SYS

การไม่สนใจขอบเขตของบรรทัด

แม้ว่าตามปกติจะใช้ข้อความสั่ง Print # เขียนบรรทัดเพียงบรรทัดเดียว หรือส่วนหนึ่งของบรรทัด แต่เราสามารถใช้อำนาจคำสั่งนี้เขียนออกมาหลาย ๆ บรรทัดในครั้งเดียว เช่นที่ทำได้ในโปรแกรม Configurator ในทางตรงกันข้าม ข้อความสั่ง Line Input # จะถูกจำกัดให้ประมวลการรับเข้าเพียงทีละหนึ่งบรรทัด

อย่างไรก็ตาม เราสามารถใช้ฟังก์ชัน Input\$ ของ Visual Basic อ่านไฟล์รับเข้า (input file) ทีละหนึ่งอักขระ ฟังก์ชัน Input\$ จะส่งคืนสตริงที่อ่านจากไฟล์ ซึ่งรวมทั้งอักขระปิดแควร์และอักขระป้อนบรรทัด วากยสัมพันธ์เป็นดังต่อไปนี้

```
Input$(charCount #fileNumber)
```

เราสามารถระบุให้อ่านตั้งแต่ 1 จนถึง 32,767 อักขระ (charCount) ภายในครั้งเดียว ถ้าไฟล์มีอักขระน้อยกว่าจำนวนที่ระบุไว้ Visual Basic จะก่อให้เกิดข้อความระบุนความผิดพลาด

Input past end of file

เราสามารถเขียนโปรแกรม Configurator ให้ใช้ฟังก์ชัน Input\$ ได้โดยง่าย ต่อไปนี้เป็นกระบวนการงานอีเวนต์ btnOpen__Click ซึ่งได้เขียนใหม่โดยใช้ Input\$

```
Sub btnOpen__Click()
```

```
Dim CLine As String
```

```
if ConfigFound Then
```

```
Beep
```

```
Exit Sub
```

```
End if
```

```
Open "C:\CONFIG.SYS" For Input As #1
```

```
txtContent.Text = Input$(LOF(1), #1) ' Assume that file has fewer  
' than 32,767 characters
```

```
Close # 1
```

```
ConfigFound = true
```

```
End Sub
```

กระบวนการงานนี้จะอ่านไฟล์ CONFIG.SYS ทั้งหมดภายในครั้งเดียว ซึ่งสามารถใช้ได้ก็เพราะว่า CONFIG.SYS เป็นไฟล์ที่มีขนาดค่อนข้างเล็กมาก ในกรณีของไฟล์ยาว ๆ จำเป็นต้องใช้ลูปเพื่อตรวจสอบการสิ้นสุดของไฟล์ (EOR) สังเกตดูกระบวนการงานที่เขียนขึ้นใหม่ข้างต้นนี้จะเห็นว่าเป็นเอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ใช้ฟังก์ชัน LOF ซึ่งแทน length of file (ความยาวของไฟล์) ฟังก์ชันนี้จะรับหมายเลขไฟล์เป็นพารามิเตอร์และส่งคืนจำนวนไบต์ในไฟล์

การทดลองเกี่ยวกับ Random File

แอปพลิเคชันฐานข้อมูล Weasel

ในตอนนี้มาเริ่มต้นสร้างแอปพลิเคชันฐานข้อมูล Weasel ซึ่งจะบันทึกเกี่ยวกับ Bavarian Tufted Forest weasels ในกรณีที่ไม่มีวีเซลก็อาจดัดแปลงโปรแกรมนี้เพื่อใช้บันทึกข้อมูลชนิดอื่นๆ

แผนพื้นฐานสำหรับฐานข้อมูล Weasel เกี่ยวข้องกับการใช้ไฟล์เข้าถึงเชิงสุ่มเก็บสารสนเทศเกี่ยวกับวีเซล เมื่อคุณดำเนินงานแอปพลิเคชัน โปรแกรมจะตรวจหรือสแกนไฟล์รวบรวมชื่อของวีเซลแต่ละตัว และแสดงรายการของชื่อในลิสต์บ็อกซ์ เมื่อผู้ใช้เลือกชื่อจากลิสต์บ็อกซ์ โปรแกรมจะค้นหาเรคคอร์ดนั้นของวีเซลในไฟล์ และแสดงข้อมูลที่ตรงประเด็นกันออกมาทั้งหมด นอกจากนี้ปุ่มคำสั่งยังทำให้สามารถเพิ่มเติมวีเซลตัวใหม่เข้าไปในไฟล์ข้อมูลหรือกำจัดวีเซลออกจากไฟล์

เริ่มต้นด้วยการสร้างโครงการใหม่ขึ้นหนึ่งโครงการ วางคอมโบบ็อกซ์หนึ่งบ็อกซ์ เท็กซ์บ็อกซ์หกบ็อกซ์ เลเบลเจ็ดเลเบล และปุ่มคำสั่งสองปุ่มแบบฟอร์ม ถ้าโครงแรกเริ่มได้แสดงไว้ในรูปที่ 4.29

รูปที่ 4.29

การออกแบบแบบฟอร์มแรกเริ่มสำหรับแอปพลิเคชันฐานข้อมูล Weasel

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ตั้งคุณสมบัติของออบเจ็กต์จอภาพตามที่กำหนดไว้ในตารางที่ 4.4 เมื่อทำแล้วแบบฟอร์มควรมีลักษณะคล้ายกับที่แสดงในรูปที่ 4.30

ออบเจ็กต์	คุณสมบัติ	ค่ากำหนด
Form1	BorderStyle	3 - Fixed Double
	Caption	Weasels
	Name	Weasels
Label1	Alignment	1 - Right Justify
	Caption	Name:
Combo1	Name	cboName
	Style	2 - Cropdown list
Label12	Alignment	1 - Right Justify
	Caption	Color:

ตารางที่ 4.4
การตั้งคุณสมบัติแอปพลิเคชันฐานข้อมูล Weasel

ออบเจ็กต์	คุณสมบัติ	ค่ากำหนด
Text1	Name	txtColor
	Text	[none]
Label13	Alignment	1 - Right Justify
	Caption	Weight:
Text2	Name	txtWeight
	Text	[none]
Label14	Alignment	1 - Right Justify
	Caption	Length:
Text3	Name	txtLength
	Text	[none]
Label15	Alignment	1 - Right Justify
	Caption	Birthdate:
Text4	Name	txtBirthdate

ตารางที่ 4.4
การตั้งคุณสมบัติแอปพลิเคชันฐานข้อมูล Weasel

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้าไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

ออบเจ็กต์	คุณสมบัติ	ค่ากำหนด
	Text	[non]
Label6	Alignment	1 - Right Justify
	Caption	Tuft Color:
Text5	Name	txtTuftColor
	Text	[none]
Label7	Alignment	1 - Right Justify
	Caption	Tuft Length:
Text6	Name	txtTuftLength
	text	[none]
Command1	Caption	Add
	Name	btnAdd
Command2	Caption	Remove
	Name	btnRemove

ตารางที่ 4.4

(ต่อ) การตั้งคุณสมบัติแอปพลิเคชันฐานข้อมูล Weasel

The screenshot shows a Windows application window titled "Form1". Inside the window, there is a data entry form. At the top, there is a label "Name" followed by a dropdown menu labeled "cboName" and a small arrow icon. Below this, there is a table with six rows. Each row has a label on the left and a text box on the right. The labels are "Color", "Weight", "Length", "Birthdate", "Tuft Color", and "Tuft Length". The text boxes are labeled "Text1", "Text2", "Text3", "Text4", "Text5", and "Text6" respectively. At the bottom of the form, there are two buttons: "Add" and "Remove".

รูปที่ 4.30

การออกแบบแบบฟอร์มที่สมบูรณ์สำหรับแอปพลิเคชันฐานข้อมูล Weasel

แอปพลิเคชันนี้ต้องการการประกาศแบบครอบคลุม เนื่องจากไม่สามารถประกาศตัวแปร ครอบคลุม และการประกาศชนิดในมอดูลแบบฟอร์ม ดังนั้นจึงจำเป็นต้องสร้างมอดูลรหัส เลือกคำสั่ง New Module จากเมนู File ในวินโดว์รหัสให้เดิมการประกาศต่อไปนี้เข้าไปในตอนการประกาศทั่วไป

Type VTFWeasel

Name As String * 32

Color As String * 12

Weight As Integer

Length As Integer

Birthdate As Double

TuftColor As String * 12

TuftLength As Integer

End Type

Global Const WEASEL_RECLEN = 70

การนิยามค่าคงตัว WEASEL_RECLEN จะทำให้อ่านโปรแกรมได้ง่ายขึ้น โดยการเห็นชื่อของค่าคงตัวแทนที่จะเป็นจำนวน 70 จะเป็นเครื่องเตือนให้ทราบว่าจริง ๆ แล้วค่านั้นหมายถึงอะไร แต่เหตุผลของการนิยามค่าคงตัวที่สำคัญมากกว่าคือ สิ่งเกี่ยวกับการปรับโปรแกรมให้ทันสมัย

สมมติว่าในอนาคตจำเป็นต้องนิยามชนิด BTFWeasel ใหม่ โดยอาจจำเป็นต้องเพิ่มฟิลด์ใหม่หรืออาจเห็นว่าอักขระ 32 ตัวนั้นไม่เพียงพอสำหรับชื่อ ถ้าไม่ได้ใช้ชื่อค่าคงตัวแทนความยาวของเรคอร์ดแล้ว ก็อาจจำเป็นต้องตรวจสอบโปรแกรมทุก ๆ บรรทัด เพื่อหาจำนวน 70 และเปลี่ยนแปลงจำนวนนั้นให้ไปเป็นค่าใหม่ นอกจากนี้ยังจะต้องมั่นใจอีกว่า 70 แต่ละค่าเป็นการอ้างอิงไปยังขนาดของเรคอร์ดจริง ๆ ไม่ได้เป็นค่าอื่น ๆ การใช้ชื่อค่าคงตัวจะทำให้สามารถเพียงแต่เปลี่ยนค่าของค่าคงตัว และส่วนที่เหลือของโปรแกรมจะถูกปรับโดยทันทีเพื่อให้ทำงานได้กับขนาดของเรคอร์ดที่นิยามใหม่

ส่วนที่เหลือของการประกาศจะเป็นแบบเฉพาะที่ต่อแบบฟอร์ม ปิดวินโดว์รหัสสำหรับมอดูลรหัส และเปิดวินโดว์รหัสสำหรับแบบฟอร์ม เดิมบรรทัดต่อไปนี้เข้าไปในตอนการประกาศทั่วไป

Dim CurrentWeaselIx As Integer

'Number of current weasel

Dim WeaselCount As Integer 'Total number of weasel

Dim RecordModified As Integer 'Boolean flag

ตัวแปรเหล่านี้จะครอบครองหมายเลขดัชนี ซึ่งบ่งบอกว่าจะแสดงเรคอร์ดของวีเซลใดออกมาจะบันทึกจำนวนทั้งหมดของเรคอร์ดของวีเซลที่เก็บอยู่ในไฟล์ และจะติดตามว่าผู้ใช้ได้ตัดแปลงข้อมูลใด ๆ บนจอภาพหรือไม่ (ในโปรแกรมนี้ใช้คำย่อ Ix แทนที่คำว่า Index เพราะ Index เป็นชื่อของคุณสมบัติของ Visual Basic)

ในเวลาเริ่มต้นใช้งานระบบ โปรแกรมควรจะอ่านแต่ละเรคอร์ดในไฟล์ และบรรจุชื่อของวีเซลทั้งหมดในลิสต์บ็อกซ์ หลังจากนั้นผู้ใช้สามารถเลือกชื่อ โปรแกรมจะตอบสนองด้วยการหยิบเอาเรคอร์ดทั้งหมดสำหรับวีเซลนั้น และแสดงเรคอร์ดนั้นออกมา ถ้าผู้ใช้เปลี่ยนแปลงแก้ไขเรคอร์ด ก็จะเขียนการเปลี่ยนแปลงที่ทำนั้นกลับไปยังไฟล์

ในการเขียนโปรแกรมที่จะทำงานได้ดีกับข้อมูลในปริมาณมาก ๆ ดังนั้นโปรแกรมนี้อ่านเรคอร์ดทั้งหมดเข้าไปในอาร์เรย์ แต่จะทำงานทั้งหมดกับตัวแปรหน่วยความจำเป็นการแทนที่อาร์เรย์ไม่สามารถมีดัชนีได้มากกว่า 32,767 ค่านี้บรรจุวีเซลเป็นจำนวนมาก และที่ 70 ไบต์ต่อเรคอร์ดจะต้องการแรมมากกว่า 2 เมกะไบต์ ถ้าบันทึกเพียงทีละหนึ่งเรคอร์ดในหน่วยความจำก็จะช่วยให้สามารถลดปริมาณของหน่วยความจำที่ต้องการลงอย่างมาก โปรแกรมนี้จะให้ชื่อทั้งหมดบรรจุในหน่วยความจำพร้อมกัน แต่ถ้าชื่อโดยเฉลี่ยมีความยาวแปดหรือเก้าอักขระ สิ่งนี้ก็ไม่ควรที่จะเป็นปัญหา สมมติว่ามีไฟล์ข้อมูลปรากฏอยู่ในตำแหน่งที่กำหนดไว้บนแผ่นดิสก์แล้ว เราจะสามารถเขียนรูทีนกำหนดค่าเริ่มต้นดังต่อไปนี้ได้ (ในแอปพลิเคชันนี้ เราไม่ได้ใช้ฟังก์ชัน FileOpenDlg แต่เราจะถือเอาชื่อไฟล์และพาทที่ตายตัว สำหรับข้อมูล ชื่อพาท C:\VB จะระบุไดเรกทอรีของ Visual Basic ที่อยู่บนระบบ ขอให้แน่ใจว่าได้ใช้ชื่อพาทที่ถูกต้องกับระบบที่ใช้)

Sub WeaselInit()

Dim Weasel As BTFWeasel, W As Integer

Open 'C:\VB\WEASEL.DAT' For Random As # 1 Len =

WEASEL_RECLEN

WeaselCount = LOF(1) / WEASEL_RECLEN

For W = 1 To WeaselCount

Get # 1, W, Weasel

cboName.AddItem Weasel.Name

Next W

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

End Sub

ข้อความสั่งนี้ทำหน้าที่เปิดหรือสร้างไฟล์ขึ้น ฟังก์ชัน LOF จะส่งคืนจำนวนทั้งหมดของไบต์ในไฟล์ซึ่งจะถูกหารด้วยขนาดของเรคอร์ด เพื่อให้ได้จำนวนของเรคอร์ดทั้งหมด หลังจากนั้นอ่านเรคอร์ดที่มีอยู่แต่ละเรคอร์ด และแทรกชื่อของวีเซลแต่ละชื่อเข้าไปในลิสต์บ็อกซ์

เมื่อการกำหนดค่าเริ่มต้นเสร็จสมบูรณ์ ผู้ใช้สามารถเลือกชื่อวีเซลเพื่อแสดงข้อมูลเกี่ยวกับวีเซล และหลังจากนั้นสามารถแก้ไขข้อมูลบางส่วน กระบวนการที่แสดงในรูปที่ 4.31 จะดำเนินการตามความต้องการนี้

กระบวนการ ShowWeasel จะรับหมายเลขดัชนีเป็นพารามิเตอร์ กระบวนการนี้จะตั้ง CurrentWellx ให้เป็นค่านี้ และหิบบข้อมูลสำหรับวีเซลนั้นออก ปรับฟิลด์แสดงผลแต่ละฟิลด์ หลังจากนั้นตั้งตัวบ่งชี้บอกว่ายังไม่ได้มีการคัดแปลงเรคอร์ดนั้น กระบวนการ UpdateCurrentWeasel จะตรวจดูว่ามีการคัดแปลงเรคอร์ดหรือไม่ ถ้ายังไม่มีการคัดแปลงกระบวนการก็เพียงแค่ยุติลง ถ้าผู้ใช้คัดแปลงเรคอร์ดก็จะอ่านค่าล่าสุดของแต่ละฟิลด์ และเขียนเรคอร์ดที่เกิดขึ้นกลับลงแผ่นดิสก์

Sub ShowWeasel (ByVal lx As Integer)

Dim Weasel As BTFWeasel

Get # 1, lx + 1, Weasel

CurrentWeaselx = lx

cboName.ListIndex = lx

txtColor.Text = Weasel.Color

txtWeight.Text = Str\$(Weasel.Weight)

txtLength.Text = Str\$(Weasel.Length)

txtBirthdate.Text = Format(Weasel.Birthdate, "mm/dd/yy")

txtTuftColor.Text = Weasel.TuftColor

txtTuftLength.Text = Str\$(Weasel.TuftLength)

RecordModified = False

End Sub

Sub UpdateCurrentWeasel()

Dim Weasel As FTFWeasel

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
Sub Form_Load( )
```

```
    WeaselInit
```

```
End Sub
```

นอกจากนี้ยังต้องเชื่อมโยงการเลือกชื่อของผู้ใช้จากลิสต์บ็อกซ์เข้ากับหน่วยแสดงเรคอร์ดของโปรแกรม และต้องเชื่อมโยงการสร้างเรคอร์ดใหม่เข้ากับปุ่ม Add สองรูทีนต่อไปนี้ทำการเชื่อมต่อเหล่านี้

```
Sub cboName_Click( )
```

```
    ShowWeasel cboName. ListIndex
```

```
End Sub
```

```
Sub btnAdd_Click( )
```

```
    Dim NewName As String
```

```
    NewName = InputBox("What is the new weasel's name?")
```

```
    If NewName = "" Then
```

```
        Exit Sub
```

```
    End If
```

```
    ShowWeasel NewWeasel(NewName)
```

```
End Sub
```

การจัดเตรียมการแสดงผลเรคอร์ดออกมานั้นทำได้ง่าย เพราะการเพียงแต่นำ 1 เข้ากับดัชนีของชื่อในลิสต์บ็อกซ์จะตรงกับดัชนีของเรคอร์ดในไฟล์ ในรูทีนที่สองจะขอให้ผู้ใช้ป้อนชื่อของวีเซลตัวใหม่ เมื่อกดปุ่มเมาส์บนปุ่ม Add ถ้าไม่ได้ป้อนชื่อเข้าไปจะไม่มีการเติมเรคอร์ดถ้าผู้ใช้ป้อนชื่อ กระบวนการจะเรียก NewWeasel เพื่อสร้างเรคอร์ด และผ่านค่าดัชนีที่ฟังก์ชันนี้ส่งคืนไปให้แก่ ShowWeasel โดยตรงที่จะได้แสดงผลเรคอร์ดใหม่ออกมา

ในการทำให้แอปพลิเคชันนี้เสร็จสมบูรณ์ จะต้องทำโปรแกรมนี้อีกส่วนหนึ่งให้สนับสนุนการแก้ไขเรคอร์ดของวีเซล เท็กซ์บ็อกซ์แต่ละบ็อกซ์ต้องการรูทีนง่าย ๆ สองรูทีนคือ กระบวนการ Change ที่เอ็กซีคิวต์ข้อความสั่ง RecordModified = True และกระบวนการ LostFocus ที่ปรับเรคอร์ดโดยการเรียก UpdateCurrentWeasel รหัสสำหรับเท็กซ์บ็อกซ์แต่ละบ็อกซ์ปรากฏอยู่ในรูปที่ 4.33

```
Sub txtColor_Change( )
```

```
    RecordModified = True
```

```
End Sub
```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```

Sub txtColor_LostFocus( )
    UpdateCurrentWease
End Sub

```

```

Sub txtWeight_Change( )
    RecordModified = True
End Sub

```

```

Sub txtWeight_LostFocus( )
    UpdateCurrentWeasel
End Sub

```

```

Sub txtLength_Change( )
    RecordModified = True
End Sub

```

```

Sub txtLength_LostFocus( )
    UpdateCurrentWeasel
End Sub

```

```

Sub txtBirthdate_Change( )
    RecordModified = True
End Sub

```

```

Sub txtBirthdate_LostFocus( )
    UpdateCurrenWeasel
End Sub

```

```

Sub txtTuftColor_Change( )

```

เอกสารนี้เป็นเอกสารที่สงวนไว้สำหรับการใช้งานเพื่อการศึกษาเท่านั้น ไม่อนุญาตให้นำไปใช้ประโยชน์ด้านการค้า
ไม่ว่ากรณีใดๆทั้งสิ้น อีกทั้งห้ามมิให้ดัดแปลงเนื้อหา และต้องอ้างอิงถึงเจ้าของเอกสารทุกครั้งที่มีการนำไปใช้

```
RecordModified = True
```

```
End Sub
```

```
Sub txtTuftColor_LostFocus( )
```

```
    UpdateCurrentWeasel
```

```
End Sub
```

```
Sub txtTuftLength_Change( )
```

```
    RecordModified = True
```

```
End Sub
```

```
Sub txtTuftLength_LostFocus( )
```

```
    UpdateCurrentWeasel
```

```
End Sub
```

รูปที่ 4.38

กระบวนการงานอีเวนต์สำหรับแท็กซ์บ็อกซ์ในแอปพลิเคชันฐานข้อมูล Weasel

การเพียงแต่ตั้งตัวบ่งชี้ในกระบวนการงาน Change แทนที่จะเรียกกรูทีน UpdateCurrentWeasel โดยตรงเป็นวิธีการที่ดีที่สุด เพราะ Visual Basic จะเรียกกระบวนการงาน Change ในทุก ๆ ครั้งที่มีการเปลี่ยนแปลงเนื้อหาของแท็กซ์บ็อกซ์ เช่น ถ้าพิมพ์ข้อความ Black ในแท็กซ์บ็อกซ์ Color กระบวนการงาน Change จะถูกเรียกห้าครั้ง แต่แต่ละครั้งสำหรับการพิมพ์อักษรแต่ละตัวแต่การปรับเรคคอร์ดไม่ควรเกิดขึ้นจนกว่าค่าในแท็กซ์บ็อกซ์จะเสร็จสมบูรณ์ ดังนั้นโปรแกรมจึงเรียกกรูทีนปรับเรคคอร์ดในกระบวนการงานอีเวนต์ LostFocus เป็นการแทนที่อีเวนต์ LostFocus คืออะไร? ตัวควบคุมจอภาพใด ๆ อาจเป็น “จุดรวมของความสนใจ” ได้ในบางเวลา ในระหว่างการเอ็กซีคิวต์แอปพลิเคชัน Windows จะบันทึกจุดรวมความสนใจนี้ด้วยอีเวนต์ GotFocus ขึ้น จุดรวมความสนใจยังคงอยู่ในแท็กซ์บ็อกซ์ในขณะที่กำลังพิมพ์หลังจากนั้นถ้ากดแป้น Tab หรือกดปุ่มเมาส์บนตัวควบคุมอื่น ๆ บางตัว แท็กซ์บ็อกซ์จะรับอีเวนต์ LostFocus และตัวควบคุมใหม่จะได้รับ GotFocus ฉะนั้นเมื่อ

ดำเนินงานกระบวนการงาน *LostFocus* ของเท็กซ์บ็อกซ์ก็จะทราบได้ทันทีว่าผู้ใช้ได้เสร็จสิ้นการพิมพ์ และโปรแกรมสามารถเรียกดูที่นปรับเรคอร์ดได้อย่างปลอดภัย

ในตอนนี้แอปพลิเคชันฐานข้อมูล *Weasel* ก็พร้อมจะดำเนินงาน กดแป้น *F5* เพื่อเริ่มใช้งานโปรแกรมเนื่องจากเป็นครั้งแรกที่ดำเนินงานโปรแกรม ดังนั้นจึงไม่มีหน่วยข้อมูลอยู่ในลิสต์บ็อกซ์กดปุ่มเมาส์บนปุ่ม *Add* และป้อนข้อมูลสำหรับวีเซลสามหรือสี่ตัวเข้าไปหลังจากนั้นกดแป้น *Alt+F4* เพื่อออกจากโปรแกรม และเริ่มใช้วานโปรแกรมใหม่อีกครั้งในทันทีตามที่แสดงในรูปที่ 4.34 ในตอนนี้จะสามารถเลือกจากชื่อที่ป้อนเข้าไปในจอภาพใช้งานก่อนหน้านี้

Form1	
Name	Sigefried
Color	white
Weight	50
Length	8
Birthdate	09/13/92
Tuft Color	Brown
Tuft Length	2
Add Remove	

รูปที่ 4.34

การดำเนินงานแอปพลิเคชันฐานข้อมูล *Weasel*

บทที่ 5 บทวิจารณ์และสรุปย่อ

สรุปย่อ

1. ในขั้นแรกเนื่องจาก Project ที่ทำขึ้นนี้เป็นประเภทโปรแกรมประยุกต์ ซึ่งมีโปรแกรมประเภทเดียวกันนี้ วางขายเป็น Packet สำเร็จรูปบ้างแล้ว ในขั้นแรกจึงได้ทำการศึกษา โปรแกรมเหล่านี้ก่อน หากคุณสมบัติต่าง ๆ เพื่อจะได้เป็นแนวทางนำมาพัฒนาโปรแกรมที่จะสร้างขึ้นต่อไป ซึ่งในที่นี้ได้ศึกษา 2 ตัวด้วยกันคือ ThaiForm ซึ่งทำงานภายใต้ระบบปฏิบัติการ DOS และ FormTool Gold ทำงานภายใต้สภาวะแวดล้อม Windows แล้วรวบรวมข้อดีข้อเสียต่าง ๆ

2. ต่อมาศึกษาแบบฟอร์มต่าง ๆ ที่มีใช้ทั้งของทางราชการและเอกชน เพื่อศึกษาว่า ซอฟต์แวร์ที่เราสร้างขึ้น จะต้องมีความสมบัติอย่างไรบ้าง เพื่อพยายามให้ครอบคลุมแบบฟอร์มที่ใช้กันอยู่โดยทั่วไป พยายามให้ครอบคลุมแบบฟอร์มที่ใช้กันอยู่โดยทั่วไป การศึกษานี้ได้ใช้วิธีสุ่มเอาตัวอย่างแบบฟอร์มบางส่วน ที่พอหาได้ง่าย ที่เห็นได้ชัดและมีมากที่สุดคือ แบบฟอร์มสมัครงาน ซึ่งรวบรวมเป็น Object ได้ดังนี้

- Label เป็นส่วนหัว ของแต่ละส่วน เช่น ชื่อ นามสกุล เป็นต้น
- Text เป็นส่วนที่จะให้เรากรอกข้อความ ส่วนมากจะเป็นเส้นประหรือเส้นทึบ
- Image อาจจะเป็น Logo ของบริษัท
- Check Box เป็นส่วนที่จะให้เราขีดเครื่องหมายเพื่อเลือก เช่น ให้เลือกเพศ เป็นต้น
- Table เช่น ตารางประวัติการศึกษา การทำงาน
- Line เป็นส่วนคั่นแบ่งแยกแต่ละส่วนเพื่อความสวยงาม

3. ศึกษาการพิมพ์บน Windows ว่า เราจะควบคุม การพิมพ์เพื่อให้ตรงตำแหน่งได้อย่างไร ความละเอียดในการกำหนดพิกัด และเลือกชนิดของ Printer ได้อย่างไร ทดลองเขียนโปรแกรม เพื่อทดสอบขึ้นมา

4. ศึกษาเรื่องเกี่ยวกับฐานข้อมูล การ Access การใช้ Data Control ศึกษา ฐานข้อมูล Format ต่างที่มีใช้กันอยู่ และ วิธีการใช้งานกับ Visual Basic

5. ออกแบบโปรแกรมโดยสรุป ได้ผลออกมาว่า จะแบ่งโปรแกรม ออกเป็น 2 ส่วน คือส่วนที่สร้างแบบฟอร์ม(Form Design) และ ส่วนที่กรอกแบบฟอร์ม (Form Entry) เพื่อลดความยุ่งยากในการควบคุม

- ส่วนสร้างแบบฟอร์ม (Form design)

เลือกใช้แบบ Single Form หมายถึงในขณะที่ โปรแกรมทำงานจะเปิด Form เพื่อออกแบบฟอร์มได้เพียงหน้าต่างเดียว เพื่อลดความยุ่งยากในการ Control Object หน้าต่างที่มีในโปรแกรม ตอนเริ่มทำงานจะมีอยู่ด้วยกัน 4 หน้าต่าง คือ

- หน้าต่างหลัก เป็น Main program มี เมนู ,Tollbar เป็นส่วนควบคุมการทำงานของโปรแกรม

- หน้าต่างออกแบบ เป็นหน้าต่างที่จะใช้ในการออกแบบ แบบฟอร์ม โดยจะเลือก Object จากหน้าต่างเครื่องมือ แล้วก็มาเปลี่ยนขนาดและตำแหน่งตามต้องการ โดยจะมีไม้บรรทัดเป็น Scale ให้คอยเทียบตำแหน่ง

- หน้าต่างเครื่องมือ เป็นหน้าต่างที่เก็บ Object ต่าง ๆ ไว้คอยให้เลือกใช้ โดยการกด Button เลือก โดยในขณะที่ Object ที่มีให้เลือกมีดังนี้

- Image เป็นส่วนที่ใช้ป้อนรูปภาพ เช่น Logo ของบริษัทโดย Formatที่ใช้จะต้องเป็นนามสกุล .BMP, .WMF, .ICO, .DIB

- Line เป็นส่วนเพิ่มเติมใส่เพื่อความสวยงาม ไว้ใช้กันระหว่างส่วนต่าง ๆ

- Label เป็นส่วนที่สำคัญส่วนหนึ่ง เป็นส่วนหัว ของ Text สามารถเลือก ขนาดและรูปแบบตัวอักษรได้เหมือน Application อื่น ๆ ของ Windows และมีคุณสมบัติตัดคำขึ้นบรรทัดใหม่ด้วยในกรณีที่กำหนดความกว้างของ Label นี้ไว้ น้อยกว่าความยาวของข้อความที่ป้อนเข้าไป

- Text เป็นส่วนที่สำคัญส่วนหนึ่งเช่นกัน เป็นส่วนที่จะต้องกรอกข้อความลงไป เพราะฉะนั้นการพิมพ์ออกทางเครื่องพิมพ์จะตรงตำแหน่งหรือไม่ก็อยู่ที่ตำแหน่งของ Text box นี้เป็นสำคัญ

- Check Box เป็นส่วนที่มีไว้ให้เลือก เช่น ในการเลือกเพศ หญิงหรือชาย เป็นต้น

- Border เป็นเส้นกรอบ สามารถเลือกขนาดและสีของกรอบได้

- Rectangle เป็นส่วนเพิ่มเติม เพื่อเพิ่มความสวยงามเป็นรูป สี่เหลี่ยมผืนผ้า สามารถเลือกขนาด ความหนาและสีของสี่เหลี่ยมได้

- Square เป็นส่วนเพิ่มเติม เพื่อเพิ่มความสวยงามเป็นรูป สี่เหลี่ยมจัตุรัส สามารถเลือกขนาด ความหนาและสีของสี่เหลี่ยมได้

- Oval เป็นส่วนเพิ่มเติม เพื่อเพิ่มความสวยงามเป็นรูป วงรี สามารถเลือกขนาด ความหนาและสีของวงรีได้

- Circle เป็นส่วนเพิ่มเติม เพื่อเพิ่มความสวยงามเป็นรูป วงกลม สามารถเลือกขนาด ความหนาและสีของวงกลมได้

- หน้าต่างคุณสมบัติ ซึ่งเป็นตารางคุณสมบัติของ Object แต่ละตัวซึ่งจะไม่เหมือนกัน จำนวนไม่เท่ากัน โดยจะแสดง คุณสมบัติของ Object ที่ Active (ถูกเลือก) อยู่ และเราสามารถเปลี่ยนคุณสมบัติของ object แต่ละตัวได้ที่หน้าต่างนี้ เช่น Left,Right,Fontname,ForeColor เป็นต้น

6. กำหนดรายละเอียดของแต่ละโปรแกรมว่าจะต้องมีฟังก์ชันการทำงานอะไรบ้าง

7. เริ่มเขียน Source code และทดสอบ

8. เพิ่ม คุณสมบัติต่าง ๆ เข้าไป เช่นอ่านข้อมูลจากฐานข้อมูล และสูตรการคำนวณต่าง ๆ เป็นต้น

หนังสืออ้างอิง

1. ราบินเดอร์ ศรีกิจจาภรณ์, “คู่มือการใช้งาน วิชาลเบสิกสำหรับวินโดว”, ซีเอ็ดยูเคชั่น
2. ศรัณย์ สัมฤทธิ์เดชขจร “การพิมพ์ภาพกราฟิกด้วย VB”, วารสารไมโครคอมพิวเตอร์, ฉบับที่ 126, 2539, หน้า 207-212.
3. ศรัณย์ ไมตรีเวช “การค้าแมนเนเจอร์ของวิชาลเบสิก 4.0”, วารสารไมโครคอมพิวเตอร์, ฉบับที่ 127, 2539, หน้า 284-290.
4. ศรัณย์ ไมตรีเวช “วิชาลเบสิกกับวิศวกรรมซอฟต์แวร์”, วารสารไมโครคอมพิวเตอร์, ฉบับที่ 128, 2539, หน้า 188-195.
5. ขจร โรจนเมธินทร์ “Visual Basic : ทำความรู้จักพรินเตอร์ออฟเจ็คต์”, วารสารไมโครคอมพิวเตอร์, ฉบับที่ 128, 2539, หน้า 261-269.
6. ศรัณย์ ไมตรีเวช “พื้นฐานการติดต่อกับการค้าเบสด้วยวิชาลเบสิก 4.0”, วารสารไมโครคอมพิวเตอร์, ฉบับที่ 128, 2539, หน้า 287-294.
7. ขจร โรจนเมธินทร์ “Visual Basic : มุมมองใหม่ของ Data Access”, วารสารไมโครคอมพิวเตอร์, ฉบับที่ 129, 2539, หน้า 225-234.
8. ดุลย์ “วิชาลเบสิก : เขียนเกมตามใจฝัน ตอน การใช้เมาส์ลากออบเจ็คต์เปลี่ยนตำแหน่ง”, วารสาร Windows Magazine, ฉบับที่ 29, 2538, หน้า 176-184.
9. Ross Nelson, “Running Visual Basic for Windows”, Microsoft Press
10. Phil Feldman, Rogger Jennings, Barry Seymour, bob Eidson, Pam Palmer, Steve Gillmor, Jack pesso, “Special Edittion Using Visual Basic 3”, Programming series