ระบบลงทะเบียนเรียน

Registration System

นายธนวัฒน์ ตระกูลกิจชัย	รหัส 6706022610039	Sec2
นายทิวากร หอมจิตต์	รหัส 6706022610047	Sec2
นายชนพล ทุ่งลาด	รหัส 6706022610322	Sec2

โครงงานนี้เป็นส่วนหนึ่งของการศึกษาหลักสูตรวิศวกรรมศาสตร์บัณฑิต

สาขาวิชาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ

คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

ปีการศึกษา 2568

ลิขสิทธิ์ของมหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ

บทคัดย่อ

โครงงานนี้มีวัตถุประสงค์เพื่อพัฒนาระบบลงทะเบียนเรียน โดยใช้ภาษา Python เป็นเครื่องมือหลักในการพัฒนา เพื่อแก้ไขปัญหาการจัดการข้อมูลแบบดั้งเดิมที่ขาดความเป็นระบบ ซึ่งมักก่อให้เกิดความล่าช้า ความซ้ำซ้อน และข้อผิดพลาดในการลงทะเบียนเรียน ระบบที่พัฒนาขึ้นสามารถจัดเก็บข้อมูลนักศึกษา รายวิชา ตารางเรียน และประวัติการลงทะเบียนได้อย่างเป็นระบบและมีประสิทธิภาพ โดยเน้นการใช้งานที่ง่าย สะดวก ลดการใช้เอกสารในรูปแบบกระดาษ และเพิ่มความถูกต้องแม่นยำในการบริหารจัดการข้อมูลการเรียนการสอน พร้อมทั้งสามารถต่อยอดเป็นระบบออนไลน์ในอนาคตได้อีกด้วย นอกจากนี้ โครงงานยังเปิดโอกาสให้นักศึกษาได้ฝึกฝนทักษะด้า นการเขียนโปรแกรม การออกแบบระบบ การวิเคราะห์ปัญหา และการทำงานร่วมกันเป็นทีม ซึ่งล้วนเป็นทักษะสำคัญต่อการประกอบวิชาชีพในสายงานวิศวกรรมสารสนเทศและเครือข่าย

กิตติกรรมประกาศ

คณะผู้จัดทำโครงงาน “ระบบลงทะเบียนเรียน” ขอกราบขอบพระคุณอาจารย์ผู้สอนวิชา COMPUTER PROGRAMMING ทุกท่านเป็นอย่างยิ่ง ที่ได้ให้คำแนะนำ ถ่ายทอดความรู้ และให้การสนับสนุนตลอดระยะเวลาการดำเนินโครงงานด้วยความเอาใจใส่และเป็นกันเอง รวมถึงขอขอบคุณผู้ที่มีส่วนเกี่ยวข้องทุกท่าน ไม่ว่าจะเป็นผู้ให้ข้อมูล แหล่งอ้างอิง หรือคำปรึกษาในด้านต่าง ๆ ซึ่งมีส่วนสำคัญอย่างยิ่งที่ช่วยให้โครงงานนี้สามารถดำเนินไปได้อย่างราบรื่นและสำเร็จลุล่วงตามเป้าหมาย

คณะผู้จัดทำขอแสดงความขอบคุณอย่างจริงใจ และขอน้อมรับข้อผิดพลาดหรือข้อบกพร่องที่อาจเกิดขึ้น เพื่อจะได้นำไปปรับปรุงและพัฒนาให้ดียิ่งขึ้นในโอกาสต่อไป

คณะผู้จัดทำ

คำนำ

การจัดทำโครงงาน “ระบบลงทะเบียนเรียน” นี้เป็นส่วนหนึ่งของวิชา Computer Programming ของหลักสูตรวิศวกรรมศาสตร์บัณฑิต สาขาวิศวกรรมสารสนเทศและเครือข่าย ภาควิชาเทคโนโลยีสารสนเทศ คณะเทคโนโลยีและการจัดการอุตสาหกรรม มหาวิทยาลัยเทคโนโลยีพระจอมเกล้าพระนครเหนือ เพื่อให้นักศึกษาได้นำความรู้ที่เรียนมาทั้งหมดมาประยุกต์ใช้ในการพัฒนาโปรแกรมที่สามารถทำงานได้จริง โดยเน้นการออกแบบและเขียนโปรแกรมด้วยภาษา Python ซึ่งเป็นภาษาที่เรียนมาในวิชา Computer Programing โดยโครงงานนี้จะช่วยการคิดวิเคราะห์และการแก้ปัญหาทางเทคนิค เพื่อเตรียมความพร้อมในการประกอบอาชีพด้านวิศวกรรมสารสนเทศและเครือข่ายในอนาคต

คณะผู้จัดทำหวังว่า รายงานฉบับนี้จะเป็นประโยชน์กับผู้อ่าน หรือนักเรียน นักศึกษา ที่กำลังหาข้อมูลเรื่องนี้อยู่ หากมีข้อแนะนำหรือข้อผิดพลาดประการใด ผู้จัดทำขอน้อมรับไว้และขอ อภัยมา ณ ที่นี้ด้วย

สารบัญ

	หน้า
บทคัดย่อ	ก
กิตติกรรมประกาศ	ข
สารบัญ	ง
สารบัญภาพ	จ
สารบัญภาพ(ต่อ)	ฉ
สารบัญตาราง	ช
บทที่ 1 บทนำ	1
1.1 ความเป็นมาและความสำคัญของโครงงาน	1
1.2 วัตถุประสงค์ของโครงงาน	1
1.3 ขอบเขตของโครงงาน	1
1.4 ประโยชน์ที่คาดว่าจะได้รับ	2
1.5 เครื่องมือที่คาดว่าจะใช้	3
บทที่ 2 ระบบลงทะเบียนเรียน	4
2.1 แฟ้มข้อมูลนักศึกษา student.dat	4
2.2 แฟ้มข้อมูลรายวิชา courses.dat	5
2.3 แฟ้มข้อมูลการลงทะเบียน enrollment.dat	5
2.4 ไฟล์ report.txt	6
บทที่ 3 การใช้งานระบบลงทะเบียนเรียน	9
3.1 การใช้งานโปรแกรมลงทะเบียนเรียน	9
บทที่ 4 อธิบายการทำงานของ Code	16
4.1 ฟังก์ชันพื้นฐานในระบบจัดการข้อมูลการลงทะเบียนเรียน	16
บทที่ 5 สรุปผลการดำเนินงานและข้อเสนอแนะ	40
5.1 สรุปผลการดำเนินงาน	40
5.2 ข้อเสนอแนะ	40
5.3 ปัญหาและการดำเนินงาน	40

สารบัญภาพ

	หน้า
ภาพที่ 2-1 ไฟล์ report.txt	7
ภาพที่ 3-1 การเลือกใช้งานฟังก์ชัน	9
ภาพที่ 3-2 ตัวเลือก Menu แต่ละฟังก์ชัน	9
ภาพที่ 3-3 Menu ฟังก์ชัน Add_Student	10
ภาพที่ 3-4 Menu ฟังก์ชัน Add_Course	10
ภาพที่ 3-5 Menu ฟังก์ชัน Add_Enrollment	10
ภาพที่ 3-6 Menu ฟังก์ชัน update_student	10
ภาพที่ 3-7 Menu ฟังก์ชัน update_course	11
ภาพที่ 3-8 Menu ฟังก์ชัน update_enrollment	11
ภาพที่ 3-9 Menu ฟังก์ชัน delete_student	11
ภาพที่ 3-10 Menu ฟังก์ชัน delete_course	11
ภาพที่ 3-11 Menu ฟังก์ชัน delete_enrollment	12
ภาพที่ 3-12 Menu ฟังก์ชัน view	12
ภาพที่ 3-13 view_student_single	12
ภาพที่ 3-14 View_student_all	13
ภาพที่ 3-15 View_student_filter	13
ภาพที่ 3-16 View_course_single	13
ภาพที่ 3-17 View_course_all	13
ภาพที่ 3-18 View_course_filter	13
ภาพที่ 3-19 View_enrollment_single	14
ภาพที่ 3-20 View_enrollment_all	14
ภาพที่ 3-21 View_enrollment_filter	14
ภาพที่ 3-22 Summary	14
ภาพที่ 3-23 report.txt	15
ภาพที่ 3-24 ออกจากโปรแกรม	15
ภาพที่ 4-1 กำหนดค่าและรูปแบบข้อมูล	16
ภาพที่ 4-2 ฟังก์ชัน str_to_bytes	19

สารบัญภาพ(ต่อ)

หน้า

ภาพที่ 4-3 ฟังก์ชัน bytes_to_str 19
ภาพที่ 4-4 ฟังก์ชัน write_record 20
ภาพที่ 4-5 ฟังก์ชัน read_all 20
ภาพที่ 4-6 ฟังก์ชัน overwrite_by_key 21
ภาพที่ 4-7 ฟังก์ชัน delete 22
ภาพที่ 4-8 ฟังก์ชัน check_file_integrity 23
ภาพที่ 4-9 ฟังก์ชัน trim_file_partial 24
ภาพที่ 4-10 ฟังก์ชัน migrate_students 25
ภาพที่ 4-11 ฟังก์ชัน CRUD – Students 26
ภาพที่ 4-12 ฟังก์ชัน add_student 26
ภาพที่ 4-13 ฟังก์ชัน update_student 27
ภาพที่ 4-14 ฟังก์ชัน delete_student 29
ภาพที่ 4-15 ฟังก์ชัน add_course 30
ภาพที่ 4-16 ฟังก์ชัน update_course 31
ภาพที่ 4-17 ฟังก์ชัน delete_course 32
ภาพที่ 4-18 ฟังก์ชัน add_enrollment 33
ภาพที่ 4-19 ฟังก์ชัน update_enrollment 34
ภาพที่ 4-20 ฟังก์ชัน delete_enrollment 35
ภาพที่ 4-21 ฟังก์ชัน generate_report() 36
ภาพที่ 4-22 view_summary() 37
ภาพที่ 4-23 ฟังก์ชัน view_filter(file, fmt, size, labels, field_idx) 37
ภาพที่ 4-24 ฟังก์ชัน view_all(file, fmt, size, labels) 38
ภาพที่ 4-25 ฟังก์ชัน init_sample_data() 38
ภาพที่ 4-26 ฟังก์ชัน main() 39

สารบัญตาราง

หน้า

ตารางที่ 2-1 แฟ้มข้อมูลนักศึกษา student.dat 4
ตารางที่ 2-2 แฟ้มข้อมูลราชวิชา courses.dat 5
ตารางที่ 2-3 แฟ้มข้อมูล Enrollment.dat 6

บทที่ 1
บทนำ

1.1 ความเป็นมาและความสำคัญของโครงงาน

ในปัจจุบัน การลงทะเบียนเรียนในสถานศึกษาหลายแห่งยังคงใช้วิธีการบันทึกข้อมูลด้วยเอกสารหรือไฟล์ทั่วไป เช่น Excel หรือแบบฟอร์มกระดาษ ซึ่งอาจทำให้เกิดปัญหาต่าง ๆ เช่น การกรอกข้อมูลซ้ำ การสูญหายของข้อมูล หรือความล่าช้าในการตรวจสอบและจัดการข้อมูลนักศึกษา ส่งผลให้การบริหารจัดการรายวิชาและการลงทะเบียนเรียนขาดความเป็นระบบ

โครงงานนี้จึงมีความสำคัญในการพัฒนา ระบบลงทะเบียนเรียนแบบง่ายด้วยภาษา Python โดยใช้ไฟล์ไบนารี (.dat) เพื่อจัดเก็บข้อมูลอย่างเป็นระเบียบและปลอดภัย ระบบสามารถเพิ่ม แก้ไข และลบข้อมูลนักศึกษาและรายวิชาได้อย่างมีประสิทธิภาพ รวมถึงรองรับฟังก์ชันการลงทะเบียนเรียน การตรวจสอบรายวิชาที่ลงทะเบียน และการสร้างรายงานสรุปข้อมูลได้ ซึ่งช่วยอำนวยความสะดวกให้แก่ผู้ใช้งาน ลดการใช้เอกสาร และส่งเสริมการบริหารจัดการข้อมูลการเรียนให้เป็นระบบมากยิ่งขึ้น

1.2 วัตถุประสงค์ของโครงงาน

1.2.1 เพื่อพัฒนาระบบลงทะเบียนเรียนได้อย่างมีประสิทธิภาพ

1.2.2 เพื่อฝึกฝนทักษะการเขียนโปรแกรมด้วยภาษา Python

1.2.3 เพื่อเรียนรู้วิธีการจัดการข้อมูลและไฟล์

1.2.4 เพื่อเรียนรู้การทำงานเป็นทีม

1.3 ขอบเขตของโครงงาน

1.3.1 ระบบยืม - ลงทะเบียนเรียนมีฟังก์ชันพื้นฐานทั้งหมด 13 ฟังก์ชัน

1.3.1.1 เพิ่มข้อมูลนักศึกษา

1.3.1.2 แก้ไขข้อมูลนักศึกษา

1.3.1.3 ลบข้อมูลนักศึกษา

1.3.1.4 ดูข้อมูลนักศึกษาแบบเจาะจง

1.3.1.5 ดูข้อมูลนักศึกษาทั้งหมด

1.3.1.6 ดูข้อมูลนักศึกษาจากการกรองรหัสนักศึกษา

1.3.1.7 เพิ่มข้อมูลรายวิชา

1.3.1.8 แก้ไขข้อมูลรายวิชา

1.3.1.9 ลบข้อมูลรายวิชา

1.3.1.10 ดูข้อมูลรายวิชาแบบเจาะจง

1.3.1.11 ดูข้อมูลรายวิชาทั้งหมด

1.3.1.12 ดูข้อมูลรายวิชาจากการกรองรหัสรายวิชา

1.3.1.13 เพิ่มข้อมูลการลงทะเบียน

1.3.1.14 แก้ไขข้อมูลการลงทะเบียน

1.3.1.15 ลบข้อมูลการลงทะเบียน

1.3.1.16 ดูข้อมูลการลงทะเบียนแบบเจาะจง

1.3.1.17 ดูข้อมูลการลงทะเบียนทั้งหมด

1.3.1.18 ดูข้อมูลการลงทะเบียนจากการกรองรหัสการลงทะเบียน

1.3.1.19 ดูข้อมูลนักศึกษา/รายวิชา/การลงทะเบียน

1.3.1.20 ดูสรุปจำนวนรวม

1.3.1.21 พิมพ์รายงาน

1.3.1.22 ออกจากระบบ

1.3.2 ระบบการยืม – คืนหนังสือห้องสมุดประกอบด้วย 4 ไฟล์ ได้แก่

1.3.2.1 แฟ้มข้อมูลนักศึก student.dat

1.3.2.2 แฟ้มข้อมูลรายวิชา courses.dat

1.3.2.3 แฟ้มข้อมูลการลงทะเบียน enrollment.dat

1.3.2.4 ไฟล์ report.txt

1.3.3 ระบบลงทะเบียนเรียนมีการจัดเก็บข้อมูลไว้ใน Text File ชื่อ report ซึ่งมี รหัสนักศึกษา ชื่อนักศึกษา สาขา จำนวนนักศึกษาในแต่ละสาขา ปีการศึกษา จำนวนนักศึกษา รหัสรายวิชา ชื่อวิชา จำนวนวิชา หน่วยกิต เกรดและสถานการณ์เรียน

1.4 ประโยชน์ที่คาดว่าจะได้รับ

1.4.1 พัฒนาระบบให้สามารถยืมและคืนหนังสือได้อย่างรวดเร็วและมีประสิทธิภาพ

1.4.2 เสริมทักษะการเขียนโปรแกรมและการแก้ไขปัญหา

1.4.3 เรียนรู้การจัดเก็บและจัดการข้อมูลในไฟล์อย่างเป็นระบบ

1.4.4 ฝึกการทำงานร่วมกับผู้อื่นในลักษณะทีมเวิร์ก

1.5 เครื่องมือที่คาดว่าจะใช้

1.6.1 โปรแกรม Visual Studio Code

1.6.2 Microsoft Office

บทที่ 2
ระบบลงทะเบียนเรียน

2.1 แฟ้มข้อมูลนักศึกษา student.dat

แฟ้มข้อมูลหนังสือประกอบด้วย 4 ฟิลด์สำคัญ แต่ละฟิลด์มีรายละเอียดและบทบาทในการจัดเก็บข้อมูลดังนี้

ฟิลด์	ชนิด	ขนาด(bytes)	ตัวอย่าง
student_id	Int	4	1001
full_name	String	50	“Somchai Dee”
year	Int	4	1
major	Int	30	“Computer Science”

ตารางที่ 2-1 แฟ้มข้อมูลนักศึกษา student.dat

2.1.1 Student_id รหัสนักศึกษา

เป็นรหัสนักศึกษาที่ใช้ในการระบุนักศึกษาแต่ละคนอย่างชัดเจนและไม่ซ้ำกันฟิลด์นี้ถูกสร้างขึ้นโดยระบบในรูปแบบของตัวเลข (integer) เช่น 1001, 1002, 1003 เป็นต้น การมีรหัสนักศึกษาที่เป็นเอกลักษณ์นี้เป็นสิ่งจำเป็นเพื่อหลีกเลี่ยงความสับสนระหว่างนักศึกษาหลายคน และช่วยให้สามารถค้นหาและเรียกดูข้อมูลของนักศึกษาได้อย่างแม่นยำและรวดเร็ว โดยเฉพาะในกรณีที่มีนักศึกษาจำนวนมาก

2.1.2 Full_name ชื่อนักศึกษา

คือ ชื่อ-นามสกุลของนักศึกษาแต่ละคน ซึ่งฟิลด์นี้จะแสดงข้อมูลชื่อ-นามสกุลแต่ละคนของมหาลัย ฟิลด์นี้เป็นประเภทข้อมูลข้อความ (string) ตัวอย่างเช่น “Somchi Dee” การมีชื่อนักศึกษาในระบบมีความสำคัญอย่างยิ่ง เพราะใช้ในการเรียกดูข้อมูล ตรวจสอบการลงทะเบียนและทำการแก้ไขข้อมูลต่างๆ นักศึกษาแต่ละคนจะมีชื่อตามที่ระบุในการลงทะเบียน และระบบจะใช้ชื่อดังกล่าวในการค้นหาและแสดงผลข้อมูลที่เกี่ยวข้องกับนักศึกษาคนนั้น

2.1.3 Year ปีการศึกษา

คือ บ่งบอกชั้นปีการศึกษาของนักศึกษา เป็นตัวเลขจำนวนเต็ม (integer) เช่น 1,2,3 หรือ 4 ช่วยให้สามารถวิเคราะห์ข้อมูลจำนวนนักศึกษาในแต่ละชั้นปีได้ง่ายขึ้น

2.1.4 major สาขาวิชา

คือ สาขาวิชาเก็บชื่อสาขาวิชาที่นักศึกษากำลังศึกษาอยู่ เป็นประเภทข้อมูลข้อความ (string) เช่น "Computer Science" หรือ "Information Technology" ใช้จัดกลุ่มนักศึกษาในรายงาน หรือคำนวณสถิติแยกตามสาขา

2.2 แฟ้มข้อมูลรายวิชา courses.dat

แฟ้มข้อมูลสมาชิกมี 3 ฟิลด์สำคัญ โดยแต่ละฟิลด์มีรายละเอียดและหน้าที่ดังต่อไปนี้

ฟิลด์	ชนิด	ขนาด(bytes)	ตัวอย่าง
course_id	Int	4	2001
course_name	String	50	"Computer Programming"
credit	Int	4	3

ตารางที่ 2-2 แฟ้มข้อมูลราชวิชา courses.dat

2.2.1 Course_id รหัสวิชา

Course_id เป็นรหัสเฉพาะของรายวิชา เช่น 2001, 2002, 2003 เป็นต้น ใช้ในการเชื่อมโยงกับไฟล์การลงทะเบียน (enrollments.dat) เพื่อบอกว่านักศึกษาเรียนวิชาอะไร

2.2.2 Course_name ชื่อวิชา

เป็นชื่อเต็มของรายวิชา เช่น "Computer Programming", "Database Systems" เป็นต้น ช่วยให้ผู้ใช้เข้าใจว่ารหัสรายวิชานั้นหมายถึงรายวิชาอะไร2.2.4

2.2.3 Credit หน่วยกิต

บ่งบอกจำนวนหน่วยกิตของรายวิชานั้น เช่น 3 หน่วยกิต ข้อมูลนี้สามารถใช้คำนวณภาระหน่วยกิตของนักศึกษาในแต่ละเทอมได้

2.3 แฟ้มข้อมูลการลงทะเบียน enrollment.dat

แฟ้มข้อมูลการยืม - คืน มี 4 ฟิลด์สำคัญ โดยแต่ละฟิลด์มีรายละเอียดและหน้าที่ดังต่อไปนี้

ฟิลด์	ชนิด	ขนาด (bytes)	ตัวอย่าง
Enroll_id	Int	4	3001
Student_id	Int	4	1001
Course_id	Int	4	2001
Grade	String	10	“A”

ตารางที่ 2-3 แฟ้มข้อมูล Enrollment.dat

2.3.1 Enroll_id รหัสการลงทะเบียน

Enroll_id เป็นรหัสเฉพาะของการลงทะเบียนแต่ละครั้ง เช่น 30001, 30002, 30003 เป็นต้น ใช้ระบุเอกลักษณ์ของข้อมูลการลงทะเบียน และเชื่อมโยงข้อมูลนักศึกษากับรายวิชาได้อย่างถูกต้อง

2.3.2 Student_id รหัสนักศึกษา

ใช้เชื่อมโยงกับแฟ้ม student.dat เพื่อระบุว่านักศึกษาคนใดลงทะเบียนเรียนในรายวิชาใด

2.3.3 Course_id รหัสวิชา

ใช้เชื่อมโยงกับแฟ้ม courses.dat เพื่อระบุรายวิชาที่นักศึกษาได้ลงทะเบียนเรียน

2.3.4 Book_ID (รหัสหนังสือ)

เป็นฟิลด์จำนวนเต็ม (integer ขนาด 5 ไบต์) ใช้เก็บรหัสหนังสือซึ่งเชื่อมโยงกับไฟล์ books.dat เพื่อบ่งบอกว่าการยืม–คืนครั้งนี้เกี่ยวข้องกับหนังสือเล่มใด

2.3.5 Grade ผลการเรียน

เก็บผลการเรียนของนักศึกษาในรายวิชานั้น เช่น “A”, “B+”, “C”, “F” เป็นต้น เป็นข้อมูลแบบข้อความ (String) ความยาวไม่เกิน 10 ตัวอักษร ใช้สำหรับรายงานผลการศึกษา

2.4 ไฟล์ report.txt

ไฟล์ report.txt ใช้เพื่อสร้างรายงานสรุปเกี่ยวกับระบบการลงทะเบียนเรียน โดยรวบรวมข้อมูลจากแฟ้ม student.dat, courses.dat และ enrollment.dat แล้วจัดทำออกมาเป็นรายงานในรูปแบบข้อความ (Text Report)

```
Library Borrow System - Summary Report
Generated At : 2025-10-02 10:55:23 (+07:00)
App Version  : 1.0
Encoding     : UTF-8

MemberID | MemberName  | BookID         | Titles                | LoanDate   | DueDate    | ReturnDate | Status
---------+-------------+----------------+----------------------+------------+------------+------------+---------
M004     | Jakkawal    | B002,B003      | -,English I           | 2025-09-30 | 2025-10-07 | 2025-09-30 | Returned
M001     | Natthanicha | B003           | English I             | 2025-09-30 | 2025-10-07 | -          | Borrowed
M003     | Phatchanoon | B001,B005,B003 | Compro,html,English I | 2025-10-01 | 2025-10-08 | 2025-10-01 | Returned

Summary (Active Books Only)
- Total Books : 5
- Active Books : 5
- Deleted Books : 0
- Borrowed Now : 1
- Available Now : 28

Borrow Statistics (Active only)
- Most Borrowed Book : English I (B003) (1 times)
- Currently Borrowed : 1
- Active Members : 3
```

ภาพที่ 2-1 ไฟล์ report.txt

2.4.1 header_text ชื่อส่วนหัวของรายงาน

เป็นฟิลด์ชนิดข้อความ (string ความยาวไม่เกิน 100 ไบต์) ใช้สำหรับเก็บชื่อของรายงาน เช่น “Registration System – Summary Report (Sample)” เพื่อให้ผู้อ่านเข้าใจได้ทันทีว่าเอกสารนี้เป็นรายงานประเภทใด หรือมีวัตถุประสงค์อย่างไร

2.4.2 generated_at วันที่และเวลาที่จัดทำรายงาน

ฟิลด์นี้เป็นข้อความ (string ขนาดไม่เกิน 25 ไบต์) ใช้สำหรับบันทึกวันและเวลาที่รายงานถูกสร้างขึ้น โดยใช้รูปแบบ "YYYY-MM-DD HH:MM" เช่น "2025-10-01 09:30" เพื่อใช้ในการอ้างอิงหรือบันทึกประวัติการสร้างรายงาน

2.4.3 app_version เวอร์ชันของแอปพลิเคชัน

ฟิลด์นี้มีขนาดข้อความไม่เกิน 10 ไบต์ ใช้เก็บหมายเลขเวอร์ชันของโปรแกรมที่สร้างรายงาน เช่น "1.0", "2.1.5" เป็นต้น เพื่อตรวจสอบว่าไฟล์รายงานนี้ถูกสร้างขึ้นจากเวอร์ชันใดของระบบ

2.4.4 encoding รูปแบบการเข้ารหัส

ฟิลด์ข้อความ (string ความยาวสูงสุด 20 ไบต์) นี้ใช้สำหรับระบุรูปแบบของการเข้ารหัสข้อมูลในรายงาน เช่น "UTF-8" หรือ "ISO8859-1" เพื่อให้แน่ใจว่าเมื่อเปิดไฟล์จะแสดงผลถูกต้องตามภาษาที่ใช้

2.4.5 ตารางข้อมูลหลัก (Main Data Table)

ส่วนนี้จะแสดงข้อมูลของนักศึกษา รายวิชา และการลงทะเบียนในรูปแบบตาราง โดยข้อมูลถูกเชื่อมโยงกันจากไฟล์ไบนารีทั้งสาม ได้แก่

StudentID | Full Name | Major | Year | CourseID | Course Name | Credit | Grade | Status StudentID / Full Name / Major / Year มาจากแฟ้ม students.dat CourseID

/ Course Name / Credit มาจากแฟ้ม courses.dat Grade / Status มาจากแฟ้ม enrollments.dat โดยที่สถานะ (Status) จะถูกกำหนดจากค่าของเกรด เช่น ถ้า Grade เป็น “W” แสดงว่า **Dropped** ฆณ์ฮถ้าเป็นเกรดอื่น เช่น “A” แสดงว่า **Active**

2.4.6 summary_section สรุปภาพรวมของข้อมูล

เป็นส่วนที่แสดงจำนวนรวมของระเบียนในแต่ละไฟล์ เช่น จำนวนนักศึกษาทั้งหมด จำนวนรายวิชา จำนวนการลงทะเบียน จำนวนรายการที่ถอนรายวิชา (Dropped Records) จำนวนที่ยังคงเรียนอยู่ (Active Records) ข้อมูลนี้ช่วยให้ผู้ดูแลระบบมองเห็นภาพรวมของฐานข้อมูลทั้งหมดได้อย่างรวดเร็ว

2.4.7 Statistics ข้อมูลสถิติผลการเรียน

เป็นส่วนที่สรุปผลการเรียนของนักศึกษาในแต่ละเกรด เช่น A, B+, C, B เป็นต้น โดยแสดงในรูปแบบนับจำนวนของนักศึกษาที่ได้เกรดนั้น ๆ (Active Only) เช่น

A count: 2

B+ count: 1

C count: 1

ข้อมูลนี้ช่วยให้ผู้ดูแลระบบสามารถวิเคราะห์ผลสัมฤทธิ์ทางการเรียนในภาพรวมของทั้งระบบได้

2.4.8 Students by Major จำนวนนักศึกษาตามสาขา

เป็นส่วนที่แสดงการแจกแจงนักศึกษาออกเป็นกลุ่มตามสาขาวิชา เช่น

Computer Science : 2

Information Technology : 2

Software Engineering : 1

ข้อมูลนี้ช่วยให้เห็นสัดส่วนนักศึกษาของแต่ละสาขาในระบบ และสามารถนำไปวิเคราะห์การกระจายตัวของนักศึกษาในระดับหลักสูตรได้

บทที่ 3
การใช้งานระบบลงทะเบียนเรียน

โปรแกรมระบบลงทะเบียนเรียนนี้ทำหน้าที่จัดการข้อมูลนักศึกษา รายวิชา และการลงทะเบียนเรียน โดยเริ่มจากการ เพิ่มข้อมูล (Add) เพื่อบันทึกข้อมูลนักศึกษา วิชา หรือการลงทะเบียนใหม่เข้าสู่ระบบ จากนั้นสามารถ แก้ไขข้อมูล (Update) เมื่อมีการเปลี่ยนแปลง เช่น แก้ชื่อสาขา หรือหน่วยกิต หากข้อมูลไม่ต้องการใช้งานแล้วสามารถ ลบข้อมูล (Delete) ออกจากระบบได้อย่างปลอดภัย ส่วนการ สร้างรายงาน (Generate Report) ใช้สรุปข้อมูลทั้งหมดในรูปแบบตาราง แสดงรายละเอียดนักศึกษา วิชา เกรด และสถานะการเรียน พร้อมสถิติรวมในไฟล์ report.txt เพื่อให้ง่ายต่อการตรวจสอบและจัดเก็บข้อมูลในภายหลัง

สำหรับผู้ใช้งานโปรแกรม

3.1 การใช้งานโปรแกรมลงทะเบียนเรียน

```
--- MAIN MENU ---
1) Add
2) Update
3) Delete
4) View
5) Generate Report
0) Exit
Choice:
```

ภาพที่ 3-1 การเลือกใช้งานฟังก์ชัน

เมื่อกดเลือกใช้งานฟังก์ชันบน Menu แล้วในแต่ละ Menu 1 จนถึง 4 จะมี Menu ย่อยอีก 3 Menu ให้เลือกใช้งาน ได้แก่ 1.Student 2.Course 3.Enrollment

```
1) Student  2) Course  3) Enrollment
Choice:
```

ภาพที่ 3-2 ตัวเลือก Menu แต่ละฟังก์ชัน

3.1.1 เมื่อเลือกกรอกหมายเลข 1 เพื่อใช้งานฟังก์ชัน add แล้ว จะสามารถเพิ่มข้อมูลของฟังก์ชัน student course Enrollment ได้

```
Choice: 1
1) Student  2) Course  3) Enrollment
Choice: 1
Student ID: 1010
Name-Surname: Thonchai Jaidee
Year: 1
Major: INE
```

ภาพที่ 3-3 Menu ฟังก์ชัน Add_Student

```
Choice: 1
1) Student  2) Course  3) Enrollment
Choice: 2
Course ID: 2010
Course Name: Network
Credit: 3
```

ภาพที่ 3-4 Menu ฟังก์ชัน Add_Course

```
Choice: 1
1) Student  2) Course  3) Enrollment
Choice: 3
Enroll ID: 30010
Student ID: 1001
Course ID: 2001
Grade: A
```

ภาพที่ 3-5 Menu ฟังก์ชัน Add_Enrollment

3.1.2 เมื่อเลือกกรอกหมายเลข 2 เพื่อใช้งานฟังก์ชัน update แล้ว จะสามารถแก้ไขข้อมูลของฟังก์ชัน student course Enrollment ได้

```
Choice: 2
1) Student  2) Course  3) Enrollment
Choice: 1
Student ID to update: 1010
Found: [1010, 'Thonchai Jaidee', 1, 'INE']
New Student ID: 1011
New Name: Thongchai Jaidee
New Year: 1
New Major: INET
Updated.
```

ภาพที่ 3-6 Menu ฟังก์ชัน update_student

```
Choice: 2
Course ID to update: 2010
Found: [2010, 'Network', 3]
New Course ID: 2011
New Course Name: Network
New Credit: 1
Updated.
```

ภาพที่ 3-7 Menu ฟังก์ชัน update_course

```
Choice: 2
1) Student  2) Course  3) Enrollment
Choice: 3
Enroll ID to update: 30010
Found: [30010, 1001, 2001, 'A']
New Enroll ID: 30011
New Student ID: 1001
New Course ID: 2001
New Grade: C-
Updated.
```

ภาพที่ 3-8 Menu ฟังก์ชัน update_enrollment

3.1.3 เมื่อเลือกกรอกหมายเลข 3 เพื่อใช้งานฟังก์ชัน delete แล้ว จะสามารถลบข้อมูลของฟังก์ชัน student course Enrollment ได้

```
Choice: 3
1) Student  2) Course  3) Enrollment
Choice: 1
Student ID to delete: 1011
Found: [1011, 'Thongchai Jaidee', 1, 'INET']
Confirm delete this student? (y/n): y
Deleted.
```

ภาพที่ 3-9 Menu ฟังก์ชัน delete_student

```
Choice: 3
1) Student  2) Course  3) Enrollment
Choice: 2
Course ID to delete: 2011
Found: [2011, 'Network', 1]
Confirm delete this course? (y/n): y
Deleted.
```

ภาพที่ 3-10 Menu ฟังก์ชัน delete_course

```
Choice: 3
1) Student  2) Course  3) Enrollment
Choice: 3
Enroll ID to delete: 30011
Found: [30011, 1001, 2001, 'C-']
Confirm delete this enrollment? (y/n): y
Deleted.
```

ภาพที่ 3-11 Menu ฟังก์ชัน delete_enrollment

3.1.4 ฟังก์ชัน view

เมื่อกรอกหมายเลข 4 เพื่อใช้งานฟังก์ชันแล้ว ภายในฟังก์ชันจะมีทั้งหมด 3 menu ให้เลือกใช้งาน ได้แก่ 1.Student 2.Course 3.Enrollment

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice:
```

ภาพที่ 3-12 Menu ฟังก์ชัน view

3.1.4.1 Student

Menu Student เมื่อเลือกใช้งาน menu นี้ ภายในจะมีให้เลือกทั้งหมด 3 menu คือ 1.Single 2.All 3.Filter

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 1
1) Single  2) All  3) Filter
Choice: 1
Enter ID: 1001
ID=1001 | Name=Somchai Dee | Year=1 | Major=Computer Science
```

ภาพที่ 3-13 view_student_single

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 1
1) Single  2) All  3) Filter
Choice: 2
[0] ID=1001 | Name=Somchai Dee | Year=1 | Major=Computer Science
[1] ID=1002 | Name=Anong Sookjai | Year=2 | Major=Information Technology
[2] ID=1003 | Name=Janpen Rungruang | Year=3 | Major=Computer Science
[3] ID=1004 | Name=Krit Prompong | Year=1 | Major=Software Engineering
[4] ID=1005 | Name=Suda Chaiyasit | Year=4 | Major=Information Technology
```

ภาพที่ 3-14 View_student_all

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 1
1) Single  2) All  3) Filter
Choice: 3
Filter keyword: 2
[1] ID=1002 | Name=Anong Sookjai | Year=2 | Major=Information Technology
```

ภาพที่ 3-15 View_student_filter

3.1.4.2 Course

Menu Course เมื่อเลือกใช้งาน menu นี้ ภายในจะมีให้เลือกทั้งหมด 3 menu คือ 1.Single 2.All 3.Filter

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 2
1) Single  2) All  3) Filter
Choice: 1
Enter ID: 2001
ID=2001 | Name=Computer Programming (Python) | Credit=3
```

ภาพที่ 3-16 View_course_single

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 2
1) Single  2) All  3) Filter
Choice: 2
[0] ID=2001 | Name=Computer Programming (Python) | Credit=3
[1] ID=2002 | Name=Data Structures | Credit=3
[2] ID=2003 | Name=Database Systems | Credit=3
[3] ID=2004 | Name=Operating Systems | Credit=3
```

ภาพที่ 3-17 View_course_all

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 2
1) Single  2) All  3) Filter
Choice: 3
Filter keyword: 4
[3] ID=2004 | Name=Operating Systems | Credit=3
```

ภาพที่ 3-18 View_course_filter

3.1.4.3 Enrollment

Menu Enrollment เมื่อเลือกใช้งาน menu นี้ ภายในจะมีให้เลือกทั้งหมด 3 menu คือ 1.Single 2.All 3.Filter

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 3
1) Single  2) All  3) Filter
Choice: 1
Enter EID: 30001
EID=30001 | StuID=1001 | CourseID=2001 | Grade=A
```

ภาพที่ 3-19 View_enrollment_single

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 3
1) Single  2) All  3) Filter
Choice: 2
[0] EID=30001 | StuID=1001 | CourseID=2001 | Grade=A
[1] EID=30002 | StuID=1001 | CourseID=2002 | Grade=B+
[2] EID=30003 | StuID=1002 | CourseID=2003 | Grade=C
[3] EID=30004 | StuID=1003 | CourseID=2001 | Grade=W
[4] EID=30005 | StuID=1004 | CourseID=2002 | Grade=A
[5] EID=30006 | StuID=1005 | CourseID=2004 | Grade=B
```

ภาพที่ 3-20 View_enrollment_all

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 3
1) Single  2) All  3) Filter
Choice: 3
Filter keyword: 01
[0] EID=30001 | StuID=1001 | CourseID=2001 | Grade=A
```

ภาพที่ 3-21 View_enrollment_filter

3.1.4.3 Summary

```
Choice: 4
1) Student  2) Course  3) Enrollment  4) Summary
Choice: 4
Students=6, Courses=5, Enrollments=7
```

ภาพที่ 3-22 Summary

3.1.6 การสร้างรายงาน (generate_report)

เมื่อเลือกหมายเลข 5 ระบบจะประมวลผลและสร้างไฟล์ report.txt ซึ่งมีข้อมูลสถิติของห้องสมุด

```
Registration System – Summary Report (Sample)
Generated At : 2025-10-01 21:24:13 (+07:00)
App Version  : 1.0
Endianness   : Little-Endian
Encoding     : UTF-8 (fixed-length)

+----------+--------------------+-----------------------+------+----------+------------------------------+--------+-------+----------+
| StudentID| Full Name          | Major                 | Year | CourseID | Course Name                  | Credit | Grade | Status   |
+----------+--------------------+-----------------------+------+----------+------------------------------+--------+-------+----------+
| 1001     | Somchai Dee        | Computer Science      | 1    | 2001     | Computer Programming (Python) | 3      | A     | Active   |
|          |                    |                       |      | 2002     | Data Structures              | 3      | B+    | Active   |
| 1002     | Anong Sookjai      | Information Technology | 2    | 2003     | Database Systems             | 3      | C     | Active   |
| 1003     | Janpen Rungruang   | Computer Science      | 3    | 2001     | Computer Programming (Python) | 3      | W     | Dropped  |
| 1004     | Krit Prompong      | Software Engineering  | 1    | 2002     | Data Structures              | 3      | A     | Active   |
| 1005     | Suda Chaiyasit     | Information Technology | 4    | 2004     | Operating Systems            | 3      | B     | Active   |
+----------+--------------------+-----------------------+------+----------+------------------------------+--------+-------+----------+

Summary (Active only)
- Total Students    : 5
- Total Courses     : 4
- Total Enrollments : 6
- Dropped Records   : 1
- Active Records    : 5

Statistics (Grade, Active only)
- A count : 2
- B+ count : 1
- C count : 1
- B count : 1

Students by Major (Active only)
- Computer Science : 2
- Information Technology : 2
- Software Engineering : 1
```

ภาพที่ 3-23 report.txt

3.1.6 การออกจากระบบ (Exit)

เมื่อกดหมายเลข 0 ระบบจะปิดการทำงานทันที

```
--- MAIN MENU ---
1) Add
2) Update
3) Delete
4) View
5) Generate Report
0) Exit
Choice: 0
PS C:\Users\Thiwa>
```

ภาพที่ 3-24 ออกจากโปรแกรม

บทที่ 4
อธิบายการทำงานของ Code

4.1 ฟังก์ชันพื้นฐานในระบบจัดการข้อมูลการลงทะเบียนเรียน

โปรแกรมนี้ทำงานบนพื้นฐานการจัดเก็บข้อมูลลงในไฟล์ไบนารี โดยใช้ Fixed-Length Records (ข้อมูลแต่ละ record มีขนาดเท่ากันเสมอ) เพื่อความสะดวกในการอ่านเขียนข้อมูลด้วยโมดูล struct ของภาษา Python

4.1.1 ส่วนกำหนดค่าและรูปแบบข้อมูล

- นำเข้าโมดูลที่จำเป็น : สำหรับจัดการ struct, ไฟล์, และเวลา
- กำหนดชื่อไฟล์ : สำหรับเก็บข้อมูลนักศึกษา รายวิชา และการลงทะเบียน
- กำหนดรูปแบบการจัดเก็บข้อมูล (Struct Format) : ระบุชนิดข้อมูลและขนาดของแต่ละ field เช่น นักศึกษามี ID, Name, Year, Major
- คำนวณขนาด record แต่ละประเภท : เพื่อใช้ตรวจสอบความสมบูรณ์ของไฟล์และการอ่านเขียน

```
import struct, os, datetime

# ===== CONFIG =====
STUDENT_FILE = "students.dat"
COURSE_FILE = "courses.dat"
ENROLL_FILE = "enrollments.dat"
REPORT_FILE = "report.txt"

# Student: ID, Name(50), Year, Major(30)
STU_FMT = "<I50sI30s"
COURSE_FMT = "<I50sI"    # course_id, name(50), credit
ENROLL_FMT = "<III10s"   # enroll_id, student_id, course_id, grade(10)

STU_SIZE = struct.calcsize(STU_FMT)
COURSE_SIZE = struct.calcsize(COURSE_FMT)
ENROLL_SIZE = struct.calcsize(ENROLL_FMT)

# ถ้าไฟล์เก่าเป็น student แบบไม่มี Major ใช้ format นี้
STU_OLD_FMT = "<I50sI"
STU_OLD_SIZE = struct.calcsize(STU_OLD_FMT)
```

ภาพที่ 4-1 กำหนดค่าและรูปแบบข้อมูล

4..1.1.1 import struct, os, datetime

- struct : ใช้ในการเข้ารหัส (pack) และถอดรหัส (unpack) ข้อมูลให้อยู่ในรูปแบบ binary
- os : ใช้สำหรับตรวจสอบและจัดการไฟล์ เช่น เช็กขนาดไฟล์ มีอยู่หรือไม่
- datetime : ใช้บันทึกเวลาที่สร้างรายงาน

4.1.1.2 # ===== CONFIG =====

ส่วนนี้เป็นการกำหนดค่าคงที่ (configuration) สำหรับไฟล์ข้อมูล

4.1.1.3 STUDENT_FILE = "students.dat"

กำหนดชื่อไฟล์ที่ใช้เก็บข้อมูลนักศึกษา

4.1.1.4 COURSE_FILE = "courses.dat"

กำหนดชื่อไฟล์ที่ใช้เก็บข้อมูลรายวิชา

4.1.1.5 ENROLL_FILE = "enrollments.dat"

กำหนดชื่อไฟล์ที่ใช้เก็บข้อมูลการลงทะเบียนเรียน

4.1.1.6 REPORT_FILE = "report.txt"

กำหนดชื่อไฟล์สำหรับสร้างรายงานสรุป

4.1.1.7 # Student: ID, Name(50), Year, Major(30)

อธิบายว่าโครงสร้างข้อมูลนักศึกษามี 4 ฟิลด์ คือ

- ID (รหัสนักศึกษา – integer)
- Name (ชื่อ – string 50 ไบต์)
- Year (ชั้นปี – integer)
- Major (สาขา – string 30 ไบต์)

4.1.1.8 STU_FMT = "<I50sI30s"

กำหนดรูปแบบ (struct format) สำหรับนักศึกษา

- < : จัดเก็บแบบ Little-Endian (ลำดับไบต์จากน้อยไปมาก)
- I : unsigned int (4 bytes) สำหรับ Student ID
- 50s : สตริงแบบ bytes ความยาวคงที่ 50
- I : unsigned int (4 bytes) สำหรับชั้นปี
- 30s : สตริงแบบ bytes ความยาวคงที่ 30 สำหรับสาขา

4.1.1.9 COURSE_FMT = "<I50sI"

9. กำหนดรูปแบบข้อมูลรายวิชา

1. I : Course ID

2. 50s : ชื่อรายวิชา (50 bytes)
3. I : จำนวนหน่วยกิต

4.1.1.10 ENROLL_FMT = "<III10s"

กำหนดรูปแบบข้อมูลการลงทะเบียน

- I : Enrollment ID
- I : Student ID (อ้างอิงไปยังนักศึกษา)
- I : Course ID (อ้างอิงไปยังรายวิชา)
- 10s : เกรด (string ความยาว 10 bytes เช่น "A", "B+", "W")

4.1.1.11 STU_SIZE = struct.calcsize(STU_FMT)

คำนวณขนาดของ record นักศึกษา (ผลลัพธ์เป็นจำนวนไบต์ที่ struct ใช้จริง)

4.1.1.12 COURSE_SIZE = struct.calcsize(COURSE_FMT)

คำนวณขนาดของ record รายวิชา

4.1.1.13 ENROLL_SIZE = struct.calcsize(ENROLL_FMT)

คำนวณขนาดของ record การลงทะเบียน

4.1.1.14 # ถ้าไฟล์เก่าเป็น student แบบไม่มี Major ใช้ format นี้

คำอธิบายว่า หากไฟล์นักศึกษามีรูปแบบเดิม (ไม่มีฟิลด์สาขา) จะใช้ struct format อื่น

4.1.1.15 STU_OLD_FMT = "<I50sI"

กำหนด format เก่าสำหรับ Student (ไม่มี Major)

4.1.1.16 STU_OLD_SIZE = struct.calcsize(STU_OLD_FMT)

คำนวณขนาด record เก่าของ Student

4.1.2 ฟังก์ชัน Utility

ฟังก์ชันในหมวดนี้เป็น ฟังก์ชันพื้นฐานที่ใช้สนับสนุนการทำงานหลัก เช่น

- การแปลงข้อมูลจาก string ⟶ bytes และกลับจาก bytes ⟶ string เพื่อรองรับการเก็บข้อมูลลงไฟล์แบบ fixed-length
- การ เขียน record ลงไฟล์
- การ อ่าน record จากไฟล์ทั้งหมด
- การ แก้ไขข้อมูล (overwrite) เฉพาะ record ที่ต้องการ
- การ ลบ record โดยเขียนค่าเป็นศูนย์ (\x00)

4.1.2.1 ฟังก์ชัน str_to_bytes

ใช้แปลงข้อความ (string) ให้เป็นข้อมูลชนิด bytes ที่มีความยาวคงที่ (fixed length) หากข้อความสั้นกว่าขนาดจะเติมด้วย \x00 และหากยาวเกินจะถูกตัดออก

```
def str_to_bytes(s, size):
    return s.encode()[:size].ljust(size, b"\x00")
```

ภาพที่ 4-2 ฟังก์ชัน str_to_bytes

4.1.2.1.1 def str_to_bytes(s, size):

ประกาศฟังก์ชันรับข้อความ s และขนาด size

4.1.2.1.2 s.encode()

แปลง string ⟶ bytes (UTF-8)

4.1.2.1.3 [:size]

ตัดข้อมูล bytes ให้มีความยาวไม่เกิน size

4.1.2.1.4 .ljust(size, b"\x00")

ถ้าข้อความสั้นกว่า size จะเติม null byte (\x00) จนเต็ม

4.1.2.1.5 return ...

คืนค่าผลลัพธ์เป็น bytes ที่มีความยาวเท่ากับ size พอดี

4.1.3 ฟังก์ชัน bytes_to_str

ใช้แปลงข้อมูลจาก bytes ที่อาจมีการเติม \x00 กลับมาเป็นข้อความปกติ

```
def bytes_to_str(b):
    return b.split(b"\x00",1)[0].decode()
```

ภาพที่ 4-3 ฟังก์ชัน bytes_to_str

4.1.3.1 b.split(b"\x00",1)

แยกข้อมูลเมื่อเจอ null byte (\x00) ครั้งแรก

4.1.3.2 [0]

เลือกเฉพาะส่วนก่อนหน้า null byte (ข้อความจริง)

4.1.3.3 decode()

แปลง bytes ⟶ string (UTF-8)

4.1.3.4 return ...

คืนค่าข้อความ string

4.1.4 ฟังก์ชัน write_record

เขียน record ลงไฟล์ในโหมดต่อท้าย (append)

```
def write_record(file, fmt, rec):
    with open(file,"ab") as f:
        f.write(struct.pack(fmt,*rec))
```

ภาพที่ 4-4 ฟังก์ชัน write_record

4.1.4.1 With open(file,"ab") as f:

ปิดไฟล์ในโหมด append binary

4.1.4.2 struct.pack(fmt,*rec)

แปลง tuple rec เป็น bytes ตามรูปแบบ fmt

4.1.4.3 f.write(...)

เขียน bytes ลงไฟล์ต่อท้าย

4.1.5 ฟังก์ชัน read_all

อ่าน record ทั้งหมดจากไฟล์ แปลงเป็นข้อมูล Python และส่งคืนเป็น list

```
def read_all(file, fmt, size):
    """อ่าน record แบบปลอดภัย ข้าม partial chunk"""
    recs=[]
    if not os.path.exists(file): return recs
    with open(file,"rb") as f:
        while True:
            chunk=f.read(size)
            if not chunk: break
            if len(chunk)!=size:
                print(f"Warning: Incomplete record in {file} ({len(chunk)} bytes, expected {size}). Ignored.")
                break
            recs.append(struct.unpack(fmt,chunk))
    return recs
```

ภาพที่ 4-5 ฟังก์ชัน read_all

4.1.5.1 recs=[ ]

สร้างลิสต์เก็บผลลัพธ์

4.1.5.2 if not os.path.exists(file): return recs

ถ้าไฟล์ไม่พบ ⟶ คืนค่าลิสต์ว่าง

4.1.5.3 with open(file,"rb") as f:

เปิดไฟล์ในโหมดอ่านแบบ binary

4.1.5.4 chunk=f.read(size)

อ่านข้อมูลทีละ record ตามขนาด

4.1.5.5 if not chunk: break

ถ้าอ่านไม่ได้ (EOF) ⟶ จบลูป

4.1.5.6 if len(chunk)!=size:

ถ้าอ่านได้ไม่ครบขนาด ⟶ ไฟล์เสีย

4.1.5.7 print(...) ; break

แจ้งเตือนและหยุดอ่าน

4.1.5.8 struct.unpack(fmt,chunk)

แปลง bytes ⟶ tuple ของค่าจริง

4.1.5.9 recs.append(...)

เก็บผลลัพธ์ลงลิสต์

4.1.5.10 return recs

ส่งคืน list ของ records

4.1.6 ฟังก์ชัน overwrite_by_key

แก้ไขข้อมูลในไฟล์ โดยค้นหา record ตามคีย์ (primary key) และเขียนทับด้วย record ใหม่

```
def overwrite_by_key(file, fmt, size, key_index, key_value, new_rec):
    recs = read_all(file, fmt, size)
    with open(file, "r+b") as f:
        for idx, r in enumerate(recs):
            vals = [bytes_to_str(x) if isinstance(x, bytes) else str(x) for x in r]
            if str(vals[key_index]) == str(key_value):
                f.seek(idx*size)
                f.write(struct.pack(fmt, *new_rec))
                return True
    return False
```

ภาพที่ 4-6 ฟังก์ชัน overwrite_by_key

4.1.6.1 recs = read_all(...)

โหลดข้อมูลทั้งหมดจากไฟล์

4.1.6.1 with open(file, "r+b") as f:

เปิดไฟล์อ่าน/เขียน binary

4.1.6.2 for idx, r in enumerate(recs):

วนลูป record ทั้งหมด

4.1.6.3 vals = [...]

แปลงค่าใน record เป็น string เพื่อเปรียบเทียบ

4.1.6.4 if str(vals[key_index]) == str(key_value):

ถ้าค่า key ตรงกับที่ต้องการ

4.1.6.5 f.seek(idx*size)

เลื่อนไปยังตำแหน่ง record นั้น

4.1.6.6 f.write(struct.pack(fmt, *new_rec))

เขียนข้อมูลใหม่ทับ record เดิม

4.1.6.7 return True

แจ้งว่าแก้ไขสำเร็จ

4.1.6.8 return False

ถ้าไม่พบ record ⟶ คืน False

4.1.7 ฟังก์ชัน delete

ลบ record โดยเขียนศูนย์ (\x00) ทับ record นั้น

```
def delete(file, size, index):
    with open(file,"r+b") as f:
        f.seek(index*size)
        f.write(b"\x00"*size)
```

ภาพที่ 4-7 ฟังก์ชัน delete

with open(file,"r+b") as f:

เปิดไฟล์อ่าน/เขียน binary

f.seek(index*size)

เลื่อนไปยังตำแหน่ง record ที่ต้องการลบ

f.write(b"\x00"*size)

เขียนทับ record ด้วย null byte เต็มขนาด

4.1.8 ฟังก์ชัน Integrity และ Migration

เนื่องจากโปรแกรมนี้จัดเก็บข้อมูลลงในไฟล์แบบ fixed-length binary record ดังนั้นหากเกิดปัญหาบางอย่าง เช่น ไฟล์ถูกเขียนไม่ครบ (partial record) หรือมีการเปลี่ยนแปลงโครงสร้างข้อมูล (เช่น เพิ่มฟิลด์ใหม่) จำเป็นต้องมีฟังก์ชันสำหรับ

4.1.8.1 ฟังก์ชัน check_file_integrity

ตรวจสอบว่าไฟล์ข้อมูลมีขนาดเป็นพหุคูณของ record size หรือไม่ เพื่อป้องกันการมี record ที่ไม่สมบูรณ์ท้ายไฟล์

```
def check_file_integrity(path, rec_size):
    if not os.path.exists(path): return True
    sz=os.path.getsize(path)
    ok=(sz%rec_size==0)
    if not ok:
        print(f"File '{path}' size {sz} not multiple of {rec_size}.")
    return ok
```

ภาพที่ 4-8 ฟังก์ชัน check_file_integrity

4.1.8.1.1 def check_file_integrity(path, rec_size):

ประกาศฟังก์ชัน รับ path ไฟล์ และขนาด record

4.8.1.1.2 if not os.path.exists(path): return True

ถ้าไฟล์ไม่พบ ⟶ คืนค่า True (ถือว่า “ไม่มีปัญหา”)

4.8.1.1.3 sz=os.path.getsize(path)

อ่านขนาดไฟล์ (หน่วยเป็น bytes)

4.8.1.1.4 ok=(sz%rec_size==0)

ตรวจสอบว่าไฟล์หารด้วยขนาด record ลงตัวหรือไม่

4.8.1.1.5 if not ok: print(...)

ถ้าไม่ลงตัว ⟶ แสดงข้อความแจ้งว่ามีปัญหา

4.8.1.1.6 return ok

คืนค่า True ถ้าไฟล์ปกติ, False ถ้ามีปัญหา

4.1.8.2 ฟังก์ชัน trim_file_partial

ตัดข้อมูลท้ายไฟล์ที่ไม่สมบูรณ์ออก (partial record) โดยทำการสำรองไฟล์ก่อนทุกครั้ง

```
def trim_file_partial(path, rec_size):
    if not os.path.exists(path): return
    sz=os.path.getsize(path)
    keep=(sz//rec_size)*rec_size
    if keep==sz: return
    bak=path+".bak"
    if not os.path.exists(bak):
        os.replace(path,bak)
        print(f"Backup: {bak}")
    with open(bak,"rb") as fin, open(path,"wb") as fout:
        fout.write(fin.read(keep))
    print(f"Trimmed {path} to {keep} bytes (removed {sz-keep}).")
```

ภาพที่ 4-9 ฟังก์ชัน trim_file_partial

4.1.8.2.1 if not os.path.exists(path): return

ถ้าไฟล์ไม่มีอยู่จริง ⟶ ไม่ทำงานต่อ

4.1.8.2.2 sz=os.path.getsize(path)

อ่านขนาดไฟล์

4.1.8.2.3 keep=(sz//rec_size)*rec_size

คำนวณจำนวนไบต์ที่เป็น record สมบูรณ์ (หารลงตัว)

4.1.8.2.4 if keep==sz: return

ถ้าไฟล์มีขนาดสมบูรณ์อยู่แล้ว ⟶ ไม่ต้องทำอะไร

4.1.8.2.5 bak=path+".bak"

สร้างชื่อไฟล์สำรอง (backup)

4.1.8.2.6 if not os.path.exists(bak): os.replace(path,bak)

ถ้า backup ยังไม่มี ⟶ ย้ายไฟล์ต้นฉบับไปเป็นไฟล์ .bak

4.1.8.2.7 with open(bak,"rb") as fin, open(path,"wb") as fout:

เปิดไฟล์สำรองอ่าน และไฟล์ใหม่เขียน

4.1.8.2.8 fout.write(fin.read(keep))

อ่านเฉพาะส่วนที่สมบูรณ์ (keep bytes) แล้วเขียนลงไฟล์ใหม่

4.1.8.3 ฟังก์ชัน migrate_students

ใช้ย้ายข้อมูลจากไฟล์นักศึกษารุ่นเก่า (ที่ไม่มีฟิลด์ Major) ไปยังไฟล์ใหม่ที่มีฟิลด์ Major โดยเพิ่มค่า default major ให้โดยอัตโนมัติ

```python
def migrate_students(old_path, old_fmt, old_size, new_path, new_fmt, default_major="Undeclared"):
    if not os.path.exists(old_path): return
    bak=old_path+".bak_migrate"
    if not os.path.exists(bak):
        os.replace(old_path,bak)
        print(f"Backup original: {bak}")
    migrated=0
    with open(bak,"rb") as fin, open(new_path,"wb") as fout:
        while True:
            chunk=fin.read(old_size)
            if not chunk: break
            if len(chunk)!=old_size: break
            sid,name_b,year=struct.unpack(old_fmt,chunk)
            major_b=default_major.encode()[:30].ljust(30,b'\x00')
            new_rec=(sid,name_b,year,major_b)
            fout.write(struct.pack(new_fmt,*new_rec))
            migrated+=1
    print(f"Migrated {migrated} records to {new_path}")
```

ภาพที่ 4-10 ฟังก์ชัน migrate_students

4.1.8.3.1 if not os.path.exists(old_path): return

ถ้าไม่มีไฟล์เก่า ⟶ ไม่ต้องทำอะไร

4.1.8.3.2 bak=old_path+".bak_migrate"

กำหนดชื่อไฟล์สำรองสำหรับการ migrate

4.1.8.3.3 if not os.path.exists(bak): os.replace(old_path,bak)

ถ้ายังไม่มี backup ⟶ ย้ายไฟล์เก่าไปเก็บเป็น .bak_migrate

4.1.8.3.4 migrated=0

ตัวนับจำนวน record ที่ migrate

4.1.8.3.5 with open(bak,"rb") as fin, open(new_path,"wb") as fout:

เปิดไฟล์เก่าอ่าน และเปิดไฟล์ใหม่เขียน

4.1.8.3.6 chunk=fin.read(old_size)

อ่าน record เก่าทีละชุด

4.1.8.3.7 if not chunk: break

ถ้าอ่านไม่เจอ ⟶ จบลูป

4.1.8.3.8 if len(chunk)!=old_size: break

ถ้าอ่านไม่ครบขนาด ⟶ หยุด (ป้องกันข้อมูลเสียหาย)

4.1.8.3.9 sid,name_b,year=struct.unpack(old_fmt,chunk)

แปลง bytes ⟶ ข้อมูลนักศึกษา (รุ่นเก่า)

4.1.8.3.10 major_b=default_major.encode()[:30].ljust(30,b'\x00')

กำหนดค่า Major เริ่มต้น (Undeclared) แล้วจัดความยาวเป็น 30 bytes

4.1.8.3.11 new_rec=(sid,name_b,year,major_b)

สร้าง record ใหม่ที่มี field Major

4.1.3.8.12 fout.write(struct.pack(new_fmt,*new_rec))

เขียน record ใหม่ลงไฟล์ใหม่

4.1.83.13 migrated+=1

เพิ่มตัวนับ record ที่ migrate

4.1.9 ฟังก์ชัน CRUD – Students

CRUD (Create, Read, Update, Delete) สำหรับ นักศึกษา (Student) คือ ฟังก์ชันที่ใช้จัดการข้อมูลของนักศึกษาโดยตรง ซึ่งทำงานโดย

```
def update_student():
    sid = input("Student ID to update: ").strip()
    recs = read_all(STUDENT_FILE, STU_FMT, STU_SIZE)
    found = False
    for r in recs:
        if str(r[0]) == sid:
            print("Found:", [bytes_to_str(x) if isinstance(x, bytes) else x for x in r])
            new_sid = int(input("New Student ID: "))
            new_name = str_to_bytes(input("New Name: "),50)
            new_year = int(input("New Year: "))
            new_major = str_to_bytes(input("New Major: "),30)
            if overwrite_by_key(STUDENT_FILE, STU_FMT, STU_SIZE, 0, sid, (new_sid,new_name,new_year,new_major)):
                print("Updated.")
            else:
                print("Update failed.")
            found = True
            break
    if not found:
        print("Student not found.")
```

ภาพที่ 4-11 ฟังก์ชัน CRUD – Students

4.1.9.1 ฟังก์ชัน add_student

เพิ่มนักศึกษาใหม่เข้าสู่ระบบ โดยรับข้อมูลจากผู้ใช้ทางคีย์บอร์ด (รหัสนักศึกษา, ชื่อ-นามสกุล, ชั้นปี, สาขาวิชา) และบันทึกเป็น record ลงในไฟล์ students.dat

```
def add_student():
    sid=int(input("Student ID: "))
    name=str_to_bytes(input("Name-Surname: "),50)
    year=int(input("Year: "))
    major=str_to_bytes(input("Major: "),30)
    write_record(STUDENT_FILE,STU_FMT,(sid,name,year,major))
```

ภาพที่ 4-12 ฟังก์ชัน add_student

4.1.9.1.1 sid=int(input("Student ID: "))

แสดงข้อความ “Student ID:” รับค่าจากผู้ใช้ และแปลงเป็นจำนวนเต็ม ⟶ ใช้เป็นรหัสนักศึกษา

name=str_to_bytes(input("Name-Surname: "),50)

รับชื่อ-สกุลจากผู้ใช้ แล้วแปลงเป็น bytes ความยาวคงที่ 50 โดยใช้ str_to_bytes

year=int(input("Year: "))

รับชั้นปีการศึกษา และแปลงเป็นจำนวนเต็ม

major=str_to_bytes(input("Major: "),30)

รับชื่อสาขา และแปลงเป็น bytes ความยาว 30

write_record(STUDENT_FILE,STU_FMT,(sid,name,year,major))

บันทึกข้อมูลทั้งหมดเป็น record ลงไฟล์ students.dat ด้วยรูปแบบ struct STU_FMT

4.1.9.2 ฟังก์ชัน update_student

ค้นหานักศึกษาจากรหัสที่ผู้ใช้ป้อนเข้ามา หากพบจะอนุญาตให้ผู้ใช้กรอกข้อมูลใหม่ และเขียนทับ record เดิมในไฟล์

```
def update_student():
    sid = input("Student ID to update: ").strip()
    recs = read_all(STUDENT_FILE, STU_FMT, STU_SIZE)
    found = False
    for r in recs:
        if str(r[0]) == sid:
            print("Found:", [bytes_to_str(x) if isinstance(x, bytes) else x for x in r])
            new_sid = int(input("New Student ID: "))
            new_name = str_to_bytes(input("New Name: "),50)
            new_year = int(input("New Year: "))
            new_major = str_to_bytes(input("New Major: "),30)
            if overwrite_by_key(STUDENT_FILE, STU_FMT, STU_SIZE, 0, sid, (new_sid,new_name,new_year,new_major)):
                print("Updated.")
            else:
                print("Update failed.")
            found = True
            break
    if not found:
        print("Student not found.")
```

ภาพที่ 4-13 ฟังก์ชัน update_student

4.1.9.2.1 sid = input("Student ID to update: ").strip()

รับ Student ID ที่ต้องการแก้ไข (ในรูป string)

recs = read_all(STUDENT_FILE, STU_FMT, STU_SIZE)

อ่านข้อมูลนักศึกษาทั้งหมดจากไฟล์

found = False

ตัวแปรสำหรับตรวจสอบว่าพบ record หรือไม่

for r in recs:

วนลูปตรวจสอบนักศึกษาทุกคน

if str(r[0]) == sid:

ถ้า Student ID (field แรกใน record) ตรงกับ input

print("Found:", [...])

แสดงข้อมูลของนักศึกษาที่พบ โดยแปลง bytes เป็น string

new_sid = int(input("New Student ID: "))

รับรหัสใหม่จากผู้ใช้

new_name = str_to_bytes(input("New Name: "),50)

รับชื่อใหม่ แล้วแปลงเป็น bytes ความยาว 50

new_year = int(input("New Year: "))

รับชั้นปีใหม่

new_major = str_to_bytes(input("New Major: "),30)

รับสาขาใหม่ แล้วแปลงเป็น bytes ความยาว 30

if overwrite_by_key(...):

ใช้ overwrite_by_key เขียนทับ record ที่มี Student ID ตรงกับที่ระบุ

print("Updated.") / print("Update failed.")

แจ้งผลลัพธ์การอัปเดต

found = True; break

ระบุว่าพบแล้ว และออกจากลูป

if not found: print("Student not found.")

ถ้าไม่พบ record ใดที่ตรงกับ ID ⟶ แจ้งว่าไม่พบ

4.1.9.3 ฟังก์ชัน delete_student

ลบนักศึกษาจากไฟล์ โดยค้นหาตาม Student ID และเขียนทับ record นั้นด้วยศูนย์ (\x00)

```python
def delete_student():
    sid = int(input("Student ID to delete: "))
    recs = read_all(STUDENT_FILE, STU_FMT, STU_SIZE)
    found = False
    for idx, r in enumerate(recs):
        if r[0] == sid:
            vals = [bytes_to_str(x) if isinstance(x, bytes) else x for x in r]
            print("Found:", vals)
            confirm = input("Confirm delete this student? (y/n): ")
            if confirm.lower() == "y":
                delete(STUDENT_FILE, STU_SIZE, idx)
                print("Deleted.")
            else:
                print("Cancelled.")
            found = True
            break
    if not found:
        print("Student not found.")
```

ภาพที่ 4-14 ฟังก์ชัน delete_student

4.1.9.3.1 sid = int(input("Student ID to delete: "))

รับ Student ID ที่ต้องการลบ

recs = read_all(STUDENT_FILE, STU_FMT, STU_SIZE)

อ่านข้อมูลนักศึกษาทั้งหมดจากไฟล์

found = False

ตัวแปรตรวจสอบว่าพบหรือไม่

for idx, r in enumerate(recs):

วนลูปข้อมูลนักศึกษาพร้อมดัชนี (index)

if r[0] == sid:

ถ้า Student ID ตรงกับที่กรอกเข้ามา

vals = [bytes_to_str(...) ...]

แปลงข้อมูลใน record ให้เป็นข้อความเพื่อแสดงผล

print("Found:", vals)

แสดงข้อมูลนักศึกษาที่เจอ

confirm = input("Confirm delete this student? (y/n): ")

ถามผู้ใช้เพื่อยืนยันการลบ

if confirm.lower() == "y":

ถ้าผู้ใช้ตอบยืนยัน ⟶ ดำเนินการลบ

delete(STUDENT_FILE, STU_SIZE, idx)

เรียกฟังก์ชัน delete เพื่อลบ record ตาม index

print("Deleted.")

แจ้งว่าลบเสร็จ

else: print("Cancelled.")

ถ้าไม่ยืนยัน ⟶ ยกเลิก

found = True; break

ระบุว่าพบแล้ว และหยุดการค้นหา

if not found: print("Student not found.")

ถ้าไม่พบ record ที่ต้องการลบ ⟶ แจ้งว่าไม่พบ

4.1.10 ฟังก์ชัน CRUD – Courses

ในระบบทะเบียนเรียน รายวิชา (Course) เป็นอีกหนึ่งหน่วยข้อมูลสำคัญที่ต้องสามารถจัดการได้เช่นเดียวกับนักศึกษา ดังนั้นจึงมีฟังก์ชัน CRUD สำหรับรายวิชา

4.1.10.1 ฟังก์ชัน add_course

เพิ่มรายวิชาใหม่ โดยรับข้อมูลจากผู้ใช้ (รหัสวิชา, ชื่อวิชา, หน่วยกิต) แล้วเขียนลงไฟล์ courses.dat

```
def add_course():
    cid=int(input("Course ID: "))
    name=str_to_bytes(input("Course Name: "),50)
    credit=int(input("Credit: "))
    write_record(COURSE_FILE,COURSE_FMT,(cid,name,credit))
```

ภาพที่ 4-15 ฟังก์ชัน add_course

4.1.10.1.1 cid=int(input("Course ID: "))

รับรหัสวิชา (Course ID) และแปลงเป็นจำนวนเต็ม

4.1.10.1.2 name=str_to_bytes(input("Course Name: "),50)

รับชื่อรายวิชา และแปลงเป็น bytes ความยาว 50

4.1.10.1.3 credit=int(input("Credit: "))

รับจำนวนหน่วยกิต และแปลงเป็น int

4.1.10.1.4 write_record(COURSE_FILE,COURSE_FMT ,(cid,name,credit))

บันทึกข้อมูลรายวิชาลงไฟล์ courses.dat ด้วยรูปแบบ struct COURSE_FMT

4.1.11 ฟังก์ชัน update_course

ค้นหารายวิชาตามรหัสวิชา หากพบให้ผู้ใช้แก้ไขข้อมูลใหม่ แล้วเขียนทับ record เดิมในไฟล์

```
def update_course():
    cid = input("Course ID to update: ").strip()
    recs = read_all(COURSE_FILE, COURSE_FMT, COURSE_SIZE)
    found = False
    for r in recs:
        if str(r[0]) == cid:
            print("Found:", [bytes_to_str(x) if isinstance(x, bytes) else x for x in r])
            new_cid = int(input("New Course ID: "))
            new_name = str_to_bytes(input("New Course Name: "),50)
            new_credit = int(input("New Credit: "))
            if overwrite_by_key(COURSE_FILE, COURSE_FMT, COURSE_SIZE, 0, cid, (new_cid,new_name,new_credit)):
                print("Updated.")
            else:
                print("Update failed.")
            found = True
            break
    if not found:
        print("Course not found.")
```

ภาพที่ 4-16 ฟังก์ชัน update_course

4.1.11.1 cid=input("Course ID to update: ").strip()

รับรหัสวิชาที่ต้องการแก้ไข

4.1.11.2 recs=read_all(COURSE_FILE,COURSE_FMT,COURSE_SIZE)

โหลดข้อมูลรายวิชาทั้งหมดจากไฟล์

4.1.11.3 found=False

ตั้งค่าเริ่มต้นว่า “ยังไม่พบ”

4.1.11.3 for r in recs:

วนลูปตรวจสอบ record ทุกตัว

4.1.11.4 if str(r[0])==cid:

ถ้า Course ID ใน record ตรงกับ input

4.1.11.5 new_cid=int(input("New Course ID: "))

รับรหัสวิชาใหม่

4.1.11.6 new_name=str_to_bytes(input("New Course Name: "),50)

รับชื่อวิชาใหม่ และแปลงเป็น bytes ความยาว 50

4.1.11.7 new_credit=int(input("New Credit: "))

รับจำนวนหน่วยกิตใหม่

4.1.11.7 if overwrite_by_key(...): print("Updated.") else: print("Update failed.")

เขียนทับ record ถ้าสำเร็จ ⟶ แสดงว่า “Updated.” ถ้าไม่สำเร็จ ⟶ แสดง “Update failed.”

4.1.12 ฟังก์ชัน delete_course

ลบรายวิชาออกจากไฟล์ โดยค้นหาตาม Course ID และทำการเขียนทับด้วย null bytes (\x00)

```
def delete_course():
    cid = int(input("Course ID to delete: "))
    recs = read_all(COURSE_FILE, COURSE_FMT, COURSE_SIZE)
    found = False
    for idx, r in enumerate(recs):
        if r[0] == cid:
            vals = [bytes_to_str(x) if isinstance(x, bytes) else x for x in r]
            print("Found:", vals)
            confirm = input("Confirm delete this course? (y/n): ")
            if confirm.lower() == "y":
                delete(COURSE_FILE, COURSE_SIZE, idx)
                print("Deleted.")
            else:
                print("Cancelled.")
            found = True
            break
    if not found:
        print("Course not found.")
```

ภาพที่ 4-17 ฟังก์ชัน delete_course

4.1.12.1 cid=int(input("Course ID to delete: "))

รับ Course ID ที่ต้องการลบ

4.1.12.1.1 recs=read_all(COURSE_FILE,COURSE_FMT ,COURSE_SIZE)

โหลดข้อมูลรายวิชาทั้งหมด

4.1.12.1.2 found=False

กำหนดค่าตั้งต้นว่า “ยังไม่พบ”

4.1.12.1.3 for idx,r in enumerate(recs):

วนลูป record พร้อมตำแหน่ง index

4.1.12.1.4 if r[0]==cid:

ถ้า Course ID ตรงกับที่ระบุ

4.1.12.1.5 vals=[bytes_to_str(...) ...]

แปลงค่าภายใน record ให้อ่านได้

4.1.12.1.6 print("Found:",vals)

แสดงรายวิชาที่พบ

4.1.12.1.7 confirm=input("Confirm delete this course? (y/n): ")

ถามผู้ใช้เพื่อยืนยันการลบ

4.1.12.1.8 if confirm.lower()=="y":

ถ้าผู้ใช้ยืนยัน (ตอบ y) ⟶ ดำเนินการลบ

4.1.13 ฟังก์ชัน CRUD – Enrollments

ในระบบทะเบียนเรียน การลงทะเบียนเรียน (Enrollment) เป็นการเชื่อมโยงระหว่างนักศึกษา (Student) และรายวิชา (Course) พร้อมกับเก็บข้อมูล เกรด ที่นักศึกษาได้ในรายวิชานั้น

4.1.13.1 ฟังก์ชัน add_enrollment

ใช้เพิ่มข้อมูลการลงทะเบียนใหม่ โดยรับรหัสการลงทะเบียน, รหัสนักศึกษา, รหัสวิชา และเกรด แล้วเขียนลงไฟล์

```
def add_enroll():
    eid=int(input("Enroll ID: "))
    sid=int(input("Student ID: "))
    cid=int(input("Course ID: "))
    grade=str_to_bytes(input("Grade: "),10)
    write_record(ENROLL_FILE,ENROLL_FMT,(eid,sid,cid,grade))
```

ภาพที่ 4-18 ฟังก์ชัน add_enrollment

4.1.13.1.1 eid=int(input("Enrollment ID: "))

รับรหัสการลงทะเบียน

sid=int(input("Student ID: "))

รับรหัสนักศึกษา

cid=int(input("Course ID: "))

รับรหัสรายวิชา

grade=str_to_bytes(input("Grade: "),10)

รับเกรด และแปลงเป็น bytes ความยาว 10

write_record(ENROLL_FILE,ENROLL_FMT,(eid,sid,cid,grade))

บันทึกข้อมูลลงไฟล์ enrollments.dat

4.1.13.2 ฟังก์ชัน update_enrollment

ค้นหาการลงทะเบียนตาม Enrollment ID และอนุญาตให้ผู้ใช้แก้ไขข้อมูลใหม่

```
def update_enroll():
    eid = input("Enroll ID to update: ").strip()
    recs = read_all(ENROLL_FILE, ENROLL_FMT, ENROLL_SIZE)
    found = False
    for r in recs:
        if str(r[0]) == eid:
            print("Found:", [bytes_to_str(x) if isinstance(x, bytes) else x for x in r])
            new_eid = int(input("New Enroll ID: "))
            new_sid = int(input("New Student ID: "))
            new_cid = int(input("New Course ID: "))
            new_grade = str_to_bytes(input("New Grade: "),10)
            if overwrite_by_key(ENROLL_FILE, ENROLL_FMT, ENROLL_SIZE, 0, eid, (new_eid,new_sid,new_cid,new_grade)):
                print("Updated.")
            else:
                print("Update failed.")
            found = True
            break
    if not found:
        print("Enrollment not found.")
```

ภาพที่ 4-19 ฟังก์ชัน update_enrollment

4.1.13.2.1 eid=input("Enrollment ID to update: ").strip()

รับรหัสการลงทะเบียนที่จะอัปเดต

4.1.13.2.2recs=read_all(ENROLL_FILE,ENROLL_FMT ,ENROLL_SIZE)

โหลดข้อมูลการลงทะเบียนทั้งหมด

4.1.13.2.3 found=False

ตั้งค่าตัวแปรว่า “ยังไม่พบ”

4.1.13.2.4 for r in recs:

วนลูปตรวจสอบทุก record

4.1.13.2.5 if str(r[0])==eid:

ถ้า Enrollment ID ตรงกับ input

4.1.13.2.6 new_eid=int(input("New Enrollment ID: "))

รับรหัสการลงทะเบียนใหม่

4.1.13.2.7 new_sid=int(input("New Student ID: "))

รับรหัสนักศึกษาใหม่

4.1.13.2.8 new_cid=int(input("New Course ID: "))

รับรหัสวิชาใหม่

4.1.13.2.9 new_grade=str_to_bytes(input("New Grade: "),10)

รับเกรดใหม่ และแปลงเป็น bytes

4.1.13.3 ฟังก์ชัน delete_enrollment

ใช้ลบการลงทะเบียนออกจากไฟล์ โดยค้นหาตาม Enrollment ID และเขียน null bytes ทับ record นั้น

```
def delete_enroll():
    eid = int(input("Enroll ID to delete: "))
    recs = read_all(ENROLL_FILE, ENROLL_FMT, ENROLL_SIZE)
    found = False
    for idx, r in enumerate(recs):
        if r[0] == eid:
            vals = [bytes_to_str(x) if isinstance(x, bytes) else x for x in r]
            print("Found:", vals)
            confirm = input("Confirm delete this enrollment? (y/n): ")
            if confirm.lower() == "y":
                delete(ENROLL_FILE, ENROLL_SIZE, idx)
                print("Deleted.")
            else:
                print("Cancelled.")
            found = True
            break
    if not found:
        print("Enrollment not found.")
```

ภาพที่ 4-20 ฟังก์ชัน delete_enrollment

4.1.13.3.1 eid=int(input("Enrollment ID to delete: "))

รับ Enrollment ID ที่ต้องการลบ

4.1.13.3.2 recs=read_all(ENROLL_FILE,ENROLL_FMT ,ENROLL_SIZE)

โหลดข้อมูลการลงทะเบียนทั้งหมด

4.1.13.3.3 found=False

ตั้งค่าเริ่มต้นว่า “ยังไม่พบ”

4.1.13.3.4 for idx,r in enumerate(recs):

วนลูปตรวจสอบทุก record พร้อม index

4.1.13.3.5 if r[0]==eid:

ถ้า Enrollment ID ตรงกับที่กรอก

4.1.13.3.6 vals=[bytes_to_str(...) ...]

แปลงข้อมูล record ให้อ่านได้

4.1.13.3.7 confirm=input("Confirm delete this enrollment? (y/n):

ถามผู้ใช้เพื่อยืนยันการลบ

4.1.14 ฟังก์ชัน generate_report()

ฟังก์ชันนี้ใช้ในการสร้างรายงานสรุปข้อมูลทั้งหมดของระบบทะเบียนเรียน ได้แก่ ข้อมูลนักศึกษา รายวิชา และการลงทะเบียน โดยจะนำข้อมูลจากไฟล์ทั้งสามมาประมวลผลรวมกัน แล้วสร้างไฟล์รายงานสรุปในรูปแบบข้อความ (Text File) เพื่อให้สามารถตรวจสอบหรือพิมพ์ออกมาได้

```
def generate_report():
    stus = read_all(STUDENT_FILE, STU_FMT, STU_SIZE)
    crs = read_all(COURSE_FILE, COURSE_FMT, COURSE_SIZE)
    ens = read_all(ENROLL_FILE, ENROLL_FMT, ENROLL_SIZE)

    # Convert students, courses to dict for easy lookup
    students = {
        s[0]: {
            "id": s[0],
            "name": bytes_to_str(s[1]),
            "year": s[2],
            "major": bytes_to_str(s[3]),
        } for s in stus if s[0] != 0
    }
    courses = {
        c[0]: {
            "id": c[0],
            "name": bytes_to_str(c[1]),
            "credit": c[2],
        } for c in crs if c[0] != 0
    }

    # Build enrollments list
    enrollments = []
    for e in ens:
        if e[0] == 0: continue
        enrollments.append({
            "enroll_id": e[0],
            "student_id": e[1],
            "course_id": e[2],
            "grade": bytes_to_str(e[3])
        })

    enrollments = sorted(enrollments, key=lambda x: x["student_id"])
```

ภาพที่ 4-21 ฟังก์ชัน generate_report()

ฟังก์ชัน generate_report() ทำหน้าที่สร้างรายงานสรุปการลงทะเบียนเรียนของนักศึกษา โดยแสดงข้อมูลนักศึกษา รายวิชา เกรด สถานะการเรียน และสถิติภาพรวมในระบบ

4.1.15 ฟังก์ชัน view_summary()

```
def view_summary():
    stus=read_all(STUDENT_FILE,STU_FMT,STU_SIZE)
    crs=read_all(COURSE_FILE,COURSE_FMT,COURSE_SIZE)
    ens=read_all(ENROLL_FILE,ENROLL_FMT,ENROLL_SIZE)
    print(f"Students={len(stus)}, Courses={len(crs)}, Enrollments={len(ens)}")
```

ภาพที่ 4-22 view_summary()

4.1.15.1 stus = read_all(STUDENT_FILE, STU_FMT, STU_SIZE)

อ่านข้อมูลนักศึกษาทั้งหมดจากไฟล์ students.dat โดยใช้รูปแบบและขนาดระเบียนตามที่กำหนด

4.1.15.2 crs = read_all(COURSE_FILE, COURSE_FMT, COURSE_SIZE)

อ่านข้อมูลรายวิชาทั้งหมดจากไฟล์ courses.dat

4.1.15.3 ens = read_all(ENROLL_FILE, ENROLL_FMT, ENROLL_SIZE)

อ่านข้อมูลการลงทะเบียนทั้งหมดจากไฟล์ enrollments.dat

4.1.15.4 print(f"Students={len(stus)}, Courses={len(crs)}, Enrollments={len(ens)}")

แสดงจำนวนของแต่ละรายการ โดยใช้ len() เพื่อคำนวณจำนวนระเบียนทั้งหมดในแต่ละไฟล์

4.1.16 ฟังก์ชัน view_filter(file, fmt, size, labels, field_idx)

ฟังก์ชันนี้ใช้สำหรับค้นหาข้อมูลภายในไฟล์ตาม “คำค้น (Keyword)” ที่ผู้ใช้ระบุ โดยจะตรวจสอบเฉพาะฟิลด์ที่กำหนดไว้ เช่น ชื่อ, รหัสนักศึกษา, หรือชื่อรายวิชา แล้วแสดงเฉพาะระเบียนที่ตรงกับเงื่อนไขบางส่วน (Partial Match)

```
def view_filter(file, fmt, size, labels, field_idx):
    key = input("Filter keyword: ").strip().lower()
    matches = []
    for i, r in enumerate(read_all(file, fmt, size)):
        vals = [bytes_to_str(x) if isinstance(x, bytes) else str(x) for x in r]
        target = str(vals[field_idx]).lower()
        if key in target:  # partial match
            matches.append((i, vals))
    if not matches:
        print("No matches found.")
    else:
        for i, vals in matches:
            print(f"[{i}] " + " | ".join(f"{labels[j]}={vals[j]}" for j in range(len(vals))))
```

ภาพที่ 4-23 ฟังก์ชัน view_filter(file, fmt, size, labels, field_idx)

4.1.17 ฟังก์ชัน view_all(file, fmt, size, labels)

ฟังก์ชันนี้ใช้แสดงข้อมูลทั้งหมดจากไฟล์ที่กำหนด โดยจะอ่านทุกระเบียนจากไฟล์ แล้วแปลงให้อยู่ในรูปแบบที่มนุษย์อ่านได้ง่าย พร้อมแสดงหมายเลขลำดับของแต่ละระเบียน

```
def view_all(file, fmt, size, labels):
    recs=read_all(file,fmt,size)
    for i,r in enumerate(recs):
        vals=[bytes_to_str(x) if isinstance(x,bytes) else x for x in r]
        print(f"[{i}] "+" | ".join(f"{labels[j]}={vals[j]}" for j in range(len(vals))))
```

ภาพที่ 4-24 ฟังก์ชัน view_all(file, fmt, size, labels)

4.1.18 ฟังก์ชัน init_sample_data()

ฟังก์ชันนี้ใช้สำหรับสร้างข้อมูลตัวอย่างเริ่มต้น (Sample Data) เพื่อให้ระบบสามารถทดสอบการทำงานได้โดยไม่ต้องป้อนข้อมูลใหม่เอง โดยจะทำการลบไฟล์เก่าทั้งหมด (ถ้ามี) และเขียนข้อมูลตัวอย่างของ นักศึกษา (Students), รายวิชา (Courses) และ การลงทะเบียน (Enrollments) ลงในไฟล์ข้อมูลที่เกี่ยวข้อง

```
def init_sample_data():
    print("Initializing sample data (overwrite files)...")

    # Students
    sample_students = [
        (1001, str_to_bytes("Somchai Dee",50), 1, str_to_bytes("Computer Science",30)),
        (1002, str_to_bytes("Anong Sookjai",50), 2, str_to_bytes("Information Technology",30)),
        (1003, str_to_bytes("Janpen Rungruang",50), 3, str_to_bytes("Computer Science",30)),
        (1004, str_to_bytes("Krit Prompong",50), 1, str_to_bytes("Software Engineering",30)),
        (1005, str_to_bytes("Suda Chaiyasit",50), 4, str_to_bytes("Information Technology",30)),
    ]
    with open(STUDENT_FILE,"wb") as f:
        for s in sample_students:
            f.write(struct.pack(STU_FMT,*s))

    # Courses
    sample_courses = [
        (2001, str_to_bytes("Computer Programming (Python)",50), 3),
        (2002, str_to_bytes("Data Structures",50), 3),
        (2003, str_to_bytes("Database Systems",50), 3),
        (2004, str_to_bytes("Operating Systems",50), 3),
    ]
    with open(COURSE_FILE,"wb") as f:
        for c in sample_courses:
            f.write(struct.pack(COURSE_FMT,*c))

    # Enrollments
    sample_enrollments = [
        (30001, 1001, 2001, str_to_bytes("A",10)),
        (30002, 1001, 2002, str_to_bytes("B+",10)),
        (30003, 1002, 2003, str_to_bytes("C",10)),
        (30004, 1003, 2001, str_to_bytes("W",10)),
        (30005, 1004, 2002, str_to_bytes("A",10)),
        (30006, 1005, 2004, str_to_bytes("B",10)),
    ]
    with open(ENROLL_FILE,"wb") as f:
        for e in sample_enrollments:
            f.write(struct.pack(ENROLL_FMT,*e))

    print("Sample data written successfully.")
```

ภาพที่ 4-25 ฟังก์ชัน init_sample_data()

4.1.19 ฟังก์ชัน main()

เป็นฟังก์ชันหลักของระบบ (Main Controller) ที่ทำหน้าที่ประสานและควบคุมการทำงานของทุกส่วนในโปรแกรม ตั้งแต่การตรวจสอบไฟล์ การสร้างข้อมูลเริ่มต้น ไปจนถึงการแสดงเมนูให้ผู้ใช้เลือกดำเนินการ เช่น เพิ่มข้อมูล แก้ไข ลบ ดูข้อมูล หรือสร้างรายงาน

```
def main():
    # เช็คความสมบูรณ์ของไฟล์ students.dat
    init_sample_data()

    if not check_file_integrity(STUDENT_FILE, STU_SIZE):
        print("Students file not aligned. Run trim or migrate as needed.")

    while True:
        print("\n--- MAIN MENU ---")
        print("1) Add")
        print("2) Update")
        print("3) Delete")
        print("4) View")
        print("5) Generate Report")
        print("0) Exit")
        c=input("Choice: ")
        if c=="1":
            print("1) Student  2) Course  3) Enrollment")
            sub=input("Choice: ")
            if sub=="1": add_student()
            elif sub=="2": add_course()
            elif sub=="3": add_enroll()
        elif c=="2":
            print("1) Student  2) Course  3) Enrollment")
            sub=input("Choice: ")
            if sub=="1": update_student()
            elif sub=="2": update_course()
            elif sub=="3": update_enroll()
        elif c=="3":
            print("1) Student  2) Course  3) Enrollment")
            sub=input("Choice: ")
            if sub=="1": delete_student()
            elif sub=="2": delete_course()
            elif sub=="3": delete_enroll()
        elif c=="4":
            print("1) Student  2) Course  3) Enrollment  4) Summary")
            sub=input("Choice: ")
            if sub=="1":
                print("1) Single  2) All  3) Filter")
                v=input("Choice: ")
                if v=="1": view_single(STUDENT_FILE,STU_FMT,STU_SIZE,["ID","Name","Year","Major"])
                elif v=="2": view_all(STUDENT_FILE,STU_FMT,STU_SIZE,["ID","Name","Year","Major"])
                elif v=="3": view_filter(STUDENT_FILE,STU_FMT,STU_SIZE,["ID","Name","Year","Major"],0)
            elif sub=="2":
                print("1) Single  2) All  3) Filter")
                v=input("Choice: ")
                if v=="1": view_single(COURSE_FILE,COURSE_FMT,COURSE_SIZE,["ID","Name","Credit"])
                elif v=="2": view_all(COURSE_FILE,COURSE_FMT,COURSE_SIZE,["ID","Name","Credit"])
                elif v=="3": view_filter(COURSE_FILE,COURSE_FMT,COURSE_SIZE,["ID","Name","Credit"],0)
            elif sub=="3":
                print("1) Single  2) All  3) Filter")
                v=input("Choice: ")
                if v=="1": view_single(ENROLL_FILE,ENROLL_FMT,ENROLL_SIZE,["EID","StuID","CourseID","Grade"])
                elif v=="2": view_all(ENROLL_FILE,ENROLL_FMT,ENROLL_SIZE,["EID","StuID","CourseID","Grade"])
                elif v=="3": view_filter(ENROLL_FILE,ENROLL_FMT,ENROLL_SIZE,["EID","StuID","CourseID","Grade"],0)
            elif sub=="4":
                view_summary()
        elif c=="5":
            generate_report()
        elif c=="0":
            break
```

ภาพที่ 4-26 ฟังก์ชัน main()

บทที่ 5
สรุปผลการดำเนินงานและข้อเสนอแนะ

5.1 สรุปผลการดำเนินงาน

จากการพัฒนาโปรแกรมระบบลงทะเบียนเรียน พบว่าโปรแกรมสามารถทำงานได้ตรงตามวัตถุประสงค์ที่กำหนดไว้ ผู้ใช้สามารถเพิ่ม แก้ไข ลบ และเรียกดูข้อมูลนักศึกษา รายวิชา และการลงทะเบียนได้อย่างถูกต้องและครบถ้วน โปรแกรมสามารถจัดเก็บข้อมูลในรูปแบบไฟล์ไบนารีได้อย่างมีประสิทธิภาพ และสามารถสร้างรายงานสรุปผลในรูปแบบไฟล์ข้อความได้สำเร็จ นอกจากนี้ระบบยังช่วยลดความซ้ำซ้อนในการจัดการข้อมูล และเพิ่มความสะดวกในการตรวจสอบข้อมูลของนักศึกษาแต่ละราย ส่งผลให้การบริหารข้อมูลการลงทะเบียนเรียนมีความรวดเร็ว ถูกต้อง และเป็นระบบมากยิ่งขึ้น.ปัญหาและอุปสรรคในการดำเนินงาน

5.2 ข้อเสนอแนะ

5.3.1 เพิ่มระบบ ตรวจสอบความถูกต้องของข้อมูล (Validation) เช่น การห้ามกรอกรหัสซ้ำหรือการตรวจสอบว่ารหัสนักศึกษาและรายวิชามีอยู่จริงก่อนบันทึกข้อมูล

5.3.2 ควรเพิ่มระบบ ค้นหาและกรองข้อมูลขั้นสูง เพื่อให้ผู้ใช้สามารถค้นหานักศึกษาหรือรายวิชาได้รวดเร็วมากยิ่งขึ้น

5.3.3 ควรพัฒนาให้สามารถ เชื่อมต่อกับฐานข้อมูล (Database) เช่น MySQL หรือ SQLite เพื่อรองรับข้อมูลจำนวนมากและเพิ่มความปลอดภัยในการจัดเก็บ

5.3.4 เพิ่มฟังก์ชัน สำรองและกู้คืนข้อมูล (Backup & Restore) เพื่อป้องกันการสูญหายของข้อมูลที่สำคัญ

5.3.5 พัฒนาให้สามารถ ส่งออกรายงานในรูปแบบไฟล์ Excel หรือ PDF เพื่อสะดวกต่อการนำไปใช้งานจริงในสถาบันการศึกษา

5.3.6 ควรปรับปรุงให้รองรับการใช้งาน หลายผู้ใช้พร้อมกัน (Multi-user) เพื่อให้เจ้าหน้าที่หลายคนสามารถจัดการข้อมูลได้ในเวลาเดียวกัน

5.3 ปัญหาและการดำเนินงาน

5.3.1 ในช่วงการพัฒนาโปรแกรมพบว่า การจัดการไฟล์ข้อมูลแบบไบนารี (.dat) มีความซับซ้อน ทำให้ต้องระมัดระวังเรื่องโครงสร้างข้อมูลและขนาดของแต่ละ record เพื่อไม่ให้เกิดการอ่านหรือเขียนข้อมูลผิดตำแหน่ง

5.3.2 การออกแบบระบบให้รองรับทั้ง นักศึกษา รายวิชา และการลงทะเบียน จำเป็นต้องใช้การเชื่อมโยงข้อมูลระหว่างไฟล์หลายไฟล์ ซึ่งอาจทำให้เกิดความซับซ้อนในการอ้างอิงรหัสต่าง ๆ เช่น Student ID และ Course ID

5.3.3 โปรแกรมยังเป็นแบบ คอนโซล (Command Line) ทำให้การใช้งานอาจไม่สะดวกสำหรับผู้ใช้ทั่วไปที่ไม่คุ้นเคยกับการพิมพ์คำสั่ง

5.3.4 การตรวจสอบความถูกต้องของข้อมูลยังมีข้อจำกัด เช่น ไม่สามารถป้องกันการกรอกรหัสซ้ำ หรือการลงทะเบียนวิชาที่ไม่มีอยู่ในระบบได้อย่างสมบูรณ์

5.3.5 การสร้างรายงาน (Report) เป็นไฟล์ข้อความธรรมดา (txt) ทำให้ รูปแบบการแสดงผลไม่สวยงามมากนัก และไม่สามารถแสดงข้อมูลในเชิงสถิติขั้นสูงได้

5.3.6 การทดสอบโปรแกรมบางส่วนต้องทำซ้ำหลายครั้ง เนื่องจากไม่มีระบบกู้คืนข้อมูลหลังจากการลบหรือเขียนทับ ทำให้ต้องเริ่มต้นสร้างข้อมูลใหม่ในแต่ละครั้ง