## การออกแบบส่วนประสานกับผู้ใช้

การออกแบบส่วนประสานกับผู้ใช้ (User Interface) เป็นกระบวนการสำคัญอีกกระบวนการหนึ่งสำหรับการผลิตซอฟต์แวร์ (มีอิทธิพลต่อการตัดสินใจเลือกใช้ซอฟต์แวร์ของลูกค้า) โดยแสดงออกมาบนจอภาพคอมพิวเตอร์ด้วยรูปสัญลักษณ์ต่าง ๆ เพื่อสื่อความหมายให้ผู้ใช้ทราบว่าจะโต้ตอบกับระบบอย่างไรเพื่อให้เกิดการทำงาน

**วิศวกรซอฟต์แวร์ต้องคำนึงถึงการใช้งานส่วนประสานเหล่านี้ ให้เรียนรู้ง่าย ไม่สร้างความสับสน และเหมาะกับผู้ใช้ในทางที่จะทำให้เกิดประโยชน์จากการทำงานให้มากที่สุด**

**ส่วนประสานกับผู้ใช้ (User Interface) :**
ในที่นี้ หมายถึงส่วนติดต่อระหว่างผู้ใช้กับระบบ เพื่อการเตรียมสารสนเทศการทำงานและการนำสารสนเทศนั้นไปใช้ด้วยการโต้ตอบกับคอมพิวเตอร์ โดยสื่อที่ใช้แสดงส่วนประสานกับผู้ใช้คือ **จอภาพ**





## การออกแบบส่วนประสานกับผู้ใช้

User Interface ->
- เป็นส่วนช่วยให้ผู้ใช้โต้ตอบกับระบบ เพื่อให้เกิดการทำงานตามวัตถุประสงค์
- เป็นเสมือนเครื่องบ่งชี้ว่าการใช้งานงานซอฟต์แวร์ นั้น ยาก ง่าย หรือซับซ้อน
- เป็นกระบวนการหนึ่งที่สำคัญไม่น้อยไปกว่าส่วนอื่นของซอฟต์แวร์ โดยในปัจจุบันมักมุ่งเน้นไปที่ส่วนประสานแบบกราฟิก (Graphic User Interface : GUI)

เนื่องจากการออกแบบมีผลต่อความพึงพอใจในตัวซอฟต์แวร์ แม้จะประมวลผลได้ดีเพียงใดก็ตาม ระหว่างการออกแบบจึงควรคำนึงถึงหลักการออกแบบ จึงมีการกำหนด
**กฎเหล็กการออกแบบที่เสนอโดย Theo Mandel 3 ประการ**
1. ให้ผู้ใช้ควบคุมการทำงานบางอย่างได้
2. ลดปริมาณของสิ่งที่ผู้ใช้ต้องจดจำลง
3. ส่วนประสานที่เลือกใช้ต้องสอดคล้องกัน





## 1. ให้ผู้ใช้ควบคุมการทำงานบางอย่างได้

ทีมออกแบบต้องตระหนักเสมอว่า "**ตนไม่ใช่ผู้ใช้ระบบ แต่ลูกค้า คือผู้ใช้ที่แท้จริง**" ดังนั้นระหว่างออกแบบส่วนประสานต้องคำนึงถึงความต้องการของผู้ใช้มากที่สุด โดยมีหลักการออกแบบดังนี้

(1) ไม่ควรบังคับให้ผู้ใช้ต้องโต้ตอบกับระบบในส่วนที่ไม่จำเป็น
เช่นการสะกดคำผิด การกำหนดสัญลักษณ์พิเศษ ฯลฯ
(2) อนุญาตให้ผู้ใช้โต้ตอบกับระบบมากกว่า 1 ทาง
กำหนดให้ใช้อุปกรณ์ที่หลากหลายได้ หรือสั่งระบบได้หลายรูปแบบ
(3) อนุญาตให้ผู้ใช้สลับการทำงานและยกเลิกผลการทำงานบางอย่างได้
สลับการทำงานไปยังโปรแกรมอื่นโดยไม่เกิดผลกระทบ หรือยกเลิกงานที่ไม่พอใจได้
(4) เตรียมเครื่องมือสร้างการทำงานแบบอัตโนมัติให้กับผู้ใช้
สร้างคีย์ลัดสำหรับผู้มีทักษะสูง เพื่ออำนวยความสะดวกงานซ้ำ ๆ เช่น copy paste
(5) ไม่ควรให้ผู้ใช้ติดต่อกับระบบปฏิบัติการด้วยการพิมพ์คำสั่งโดยตรง
ควรสร้าง wizard เพื่อช่วยผู้ใช้ติดต่อกับระบบได้โดยตรง
(6) ผู้ใช้ควรทำงานกับอ็อบเจ็คได้โดยตรง
เช่น ปรับขนาดรูปภาพโดยใช้เมาส์ลากและเห็นความเปลี่ยนแปลง





## 2. ลดปริมาณของสิ่งที่ผู้ใช้ต้องจดจำลง

ซอฟต์แวร์ใดที่ผู้ใช้ต้องจดจำรายละเอียดการทำงานเองมากจนเกินไป ย่อมเสี่ยงต่อการเกิดความผิดพลาดในการใช้งานสูง ดังนั้น ซอฟต์แวร์หรือระบบควรจัดเก็บรายละเอียดเกี่ยวกับการโต้ตอบบางอย่างของผู้ใช้ไว้ เพื่อช่วยเตือนความจำในภายหลังได้ โดยยึดหลักการออกแบบดังนี้

(1) ลดการจดจำการใช้งานที่ผ่านไปขณะที่ใช้โปรแกรมนั้นอยู่
ควรออกแบบให้มีรายการแสดงสิ่งที่ผู้ใช้กระทำไป จะได้ไม่ต้องนึกย้อนกลับ
(2) ควรกำหนดค่าเริ่มต้นการใช้งานที่เหมาะสมกับผู้ใช้ทั่วไป
ควรมีตัวเลือกเริ่มต้น เช่น ตัวอักษร ขนาด รูปแบบ โดยเปลี่ยนแปลงและ ย้อนกลับได้
(3) คีย์ลัด หรือ Shortcut key ควรสื่อความหมายของงานได้ชัดเจนและจดจำง่าย
ควรมีคีย์ลัดที่ใช้งานและเป็นมาตรฐาน เช่น Ctrl-c, Ctrl-v, Ctrl-S , Ctrl-P เป็นต้น
(4) ควรแสดงสถานะการทำงานของผู้ใช้ในกระบวนการใด ๆ
ควรมีตัวแสดงสถานะที่ผ่านกระบวนไปแล้วด้วยการใช้สัญลักษณ์
(5) ควรแสดงรายละเอียดในการใช้งานพอสังเขปในเบื้องต้น
ควรมีตัวแสดงรายละเอียดที่กำหนดปัจจุบัน เป็น Font , กรอบ , สี, รูปแบบ

## 3. ส่วนประสานต้องสอดคล้องกัน

ส่วนประสานที่สอดคล้องกัน หมายถึง ส่วนประสานเหล่านั้นจะต้องเชื่อมโยงกันอย่างเป็นลำดับขั้นตอน ซึ่งต้องผ่านการออกแบบการเชื่อมโยงมาเป็นอย่างดี ตลอดจนมีส่วนนำทางการใช้งานแก่ผู้ใช้ โดยมีหลักการออกแบบดังนี้

(1) ส่วนประกอบทุกอย่างบนจอภาพของการทำงานอย่างใดอย่างหนึ่งจะต้องสอดคล้องกัน

ขณะที่กำลังทำงานหนึ่ง ๆ ทุกส่วนประกอบควรแสดงชื่องานที่สอดคล้องกัน หรือมีส่วนแสดงความเกี่ยวข้องกับงานดังกล่าว หรือแจ้งเตือนหากต้องการยกเลิกงานดังกล่าว

(2) โปรแกรมที่อยู่ในกลุ่มผลิตภัณฑ์เดียวกัน (Program Families) จะต้องมีส่วนประสานเหมือนกัน และสอดคล้องกัน

แม้ว่ามีวัตถุประสงค์แตกต่างกัน แต่การเลือกใช้โปรแกรมควรให้อยู่ผลิตภัณฑ์เดียวกัน เพราะจะได้มีส่วนประสานเหมือนกัน เข้าใจตรงกัน

(3) ไม่ควรเปลี่ยนลักษณะการโต้ตอบกับระบบที่โปรแกรมส่วนใหญ่ใช้เหมือนกัน

เนื่องจากผู้ใช้มักคุ้นเคยกับการโต้ตอบลักษณะหนึ่ง ๆ หากมีการเปลี่ยนแปลงหรือแตกต่าง ผู้ใช้อาจสับสนได้ รวมถึงการใช้งานก็ควรมีรูปแบบโต้ตอบเดียวกันทั้งหมด





## ชนิดของส่วนประสานกับผู้ใช้

เนื่องจากส่วนประสานกับผู้ใช้เป็นส่วนหนึ่งของระบบที่ช่วยให้ผู้ใช้ นำเข้าข้อมูล สั่งให้ระบบทำงาน แสดงผลลัพธ์ที่เกิดจากการทำงานของระบบ

**ดังนั้นชนิดของส่วนประสานกับผู้ใช้จึงแบ่งได้ 2 รูปแบบคือ**

- รูปแบบการโต้ตอบระหว่างผู้ใช้กับระบบ
- รูปแบบการนำเสนอสารสนเทศต่อผู้ใช้

**(1) รูปแบบการโต้ตอบระหว่างผู้ใช้กับระบบ**

คือ ลักษณะที่ผู้ใช้สั่งงานคอมพิวเตอร์ด้วยข้อมูลบางอย่าง เพื่อให้คอมพิวเตอร์ทำงานตามที่ต้องการ ในอดีตการโต้ตอบกับระบบมีเพียงรูปแบบเดียวคือ Command-Line Interface ซึ่งเป็นเรื่องยากสำหรับผู้ใช้ทั่วไป แต่ในปัจจุบันมีการพัฒนารูปแบบการโต้ตอบให้ง่ายมากยิ่งขึ้น จำแนกได้ดังนี้





## ชนิดของส่วนประสานกับผู้ใช้

**(1) รูปแบบการโต้ตอบระหว่างผู้ใช้กับระบบ**

**1.1 การโต้ตอบกับระบบโดยตรง (Direct Manipulation)**

คือ ลักษณะที่ผู้ใช้ดำเนินการกับอ็อบเจ็คบนจอภาพคอมพิวเตอร์โดยตรงผ่านอุปกรณ์นำเข้าข้อมูล (Mouse, Stylus, Trackball, Touch screen) ไปยังอ็อบเจ็คนั้นโดยตรง เช่น ใช้เมาส์ลากวัตถุไปวางยังตำแหน่งอื่น ๆ ฯลฯ

**1.2 การเลือกเมนูคำสั่ง (Menu Selection)**

คือ ลักษณะที่ผู้ใช้ต้องเลือกคำสั่งจากรายการที่โปรแกรมเตรียมไว้ให้โดยเมนูมักมี 2 ลักษณะ คือ Pull-down Menu, Pop-up Menu

**1.3 การป้อนข้อมูลลงในฟอร์ม (Form Fill-in)**

คือ ลักษณะที่ผู้ใช้ต้องป้อนข้อมูลลงในช่องว่างต่างๆ ตามหัวข้อของข้อมูล ในแบบฟอร์มป้อนข้อมูลอาจมีปุ่มคำสั่งให้ผู้ใช้คลิกเพื่อให้เกิดการทำงานต่อไป เช่น เพิ่ม ลบ แก้ไข เป็นต้น





## ชนิดของส่วนประสานกับผู้ใช้

**(1) รูปแบบการโต้ตอบระหว่างผู้ใช้กับระบบ**

**1.4 การโต้ตอบด้วยภาษาธรรมชาติ (Natural Language)**

คือ ลักษณะที่ผู้ใช้สั่งงานคอมพิวเตอร์ด้วยการเขียน/เสียง/คำพูดในภาษาต่าง ๆ โดยเครื่องคอมพิวเตอร์ต้องมีระบบแปลภาษา/เสียง/คำพูดอยู่ภายในหรือจะต้องเป็นไปตามรูปแบบตัวแปลภาษารู้จัก

| รูปแบบการโต้ตอบ | ข้อดี | ข้อเสีย |
|---|---|---|
| Direct Manipulation | รวดเร็วและสื่อความหมายได้ชัดเจน ง่ายต่อการเรียนรู้ | ยากต่อการพัฒนา |
| Menu Selection | ช่วยลดข้อผิดพลาดในการใช้งานของผู้ใช้ เนื่องจากผู้ใช้ไม่ต้องพิมพ์คำสั่งเอง | ใช้งานได้ช้ากว่าการพิมพ์คำสั่งสำหรับผู้ใช้ที่เชี่ยวชาญ และหากมีเมนูจำนวนมาก จะทำให้ดูซับซ้อน |
| Form Fill-in | ป้อนข้อมูลได้ง่าย | ต้องใช้พื้นที่ในหน้าจอมาก และเสี่ยงต่อการเกิดข้อผิดพลาด หากผู้ใช้ป้อนข้อมูลผิด |
| Natural Language | ง่ายต่อการใช้งานสำหรับผู้ใช้ทั่วไป | ต้องอาศัยกลไกลแปลภาษาที่มีประสิทธิภาพสูง |

## ชนิดของส่วนประสานกับผู้ใช้

### (2) รูปแบบการนำเสนอสารสนเทศต่อผู้ใช้

การโต้ตอบส่วนใหญ่ ก็เพื่อต้องการข้อมูลหรือสารสนเทศตามวัตถุประสงค์ของงาน ทีมงานจึงมีหน้าที่กำหนดรูปแบบการนำเสนอสารสนเทศเหล่านั้นให้ครบถ้วนและมีประสิทธิภาพ เพื่อสร้างความพึงพอใจให้กับผู้ใช้มากที่สุด โดยสารสนเทศอาจอยู่ในรูปแบบข้อความ กราฟ หรือรูปภาพ แล้วแต่ความต้องการของผู้ใช้แต่ละระดับ

★ **<u>สิ่งที่ควรคำนึงถึง</u>** คือ ต้องไม่นำส่วนข้อมูลและส่วนแสดงผลข้อมูลรวมอยู่ด้วยกัน เนื่องจากหากต้องการมีความเปลี่ยนแปลงรูปลักษณ์การนำเสนอหรือแสดงผลข้อมูลจะทำได้ยาก ควรแยกส่วนออกจากกันจะมีผลดีในอนาคต

**การนำเสนอสารสนเทศแบ่งเป็น 2 รูปแบบดังนี้**

2.1 Alphanumeric Information
2.2 Dynamically Varying Information



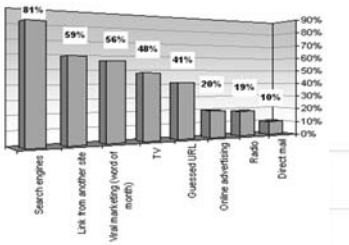


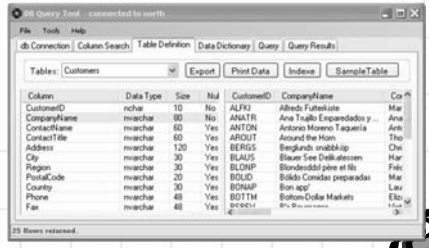
## ชนิดของส่วนประสานกับผู้ใช้

### (2) รูปแบบการนำเสนอสารสนเทศต่อผู้ใช้

**2.1 Alphanumeric Information**

คือ สารสนเทศที่ไม่เปลี่ยนแปลงตามวันหรือเวลา เป็นข้อมูลการดำเนินงานทางธุรกิจทั่วไป เช่น ข้อมูลยอดขาย ผลกำไร-ขาดทุน เป็นต้น สารสนเทศประเภทนี้ควรนำเสนอด้วยตาราง หรือกราฟ





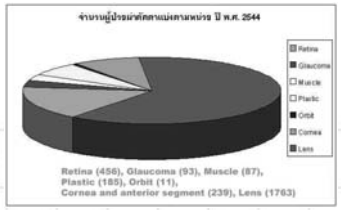


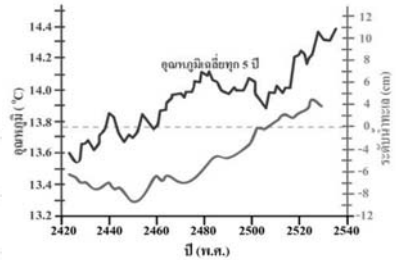
## ชนิดของส่วนประสานกับผู้ใช้

### (2) รูปแบบการนำเสนอสารสนเทศต่อผู้ใช้

**2.2 Dynamically Varying Information**

คือ สารสนเทศที่มีการเปลี่ยนแปลงขึ้น-ลงตามวันหรือเวลา เช่น อุณหภูมิระดับน้ำทะเล หรือดัชนีหุ้น เป็นต้น สารสนเทศประเภทนี้ควรนำเสนอในรูปแบบกราฟิก และควรเลือกประเภทกราฟเส้น หรือวงกลม








## การวิเคราะห์ผู้ใช้ (User Analysis)

การออกแบบส่วนประสานกับผู้ใช้ที่มีประสิทธิภาพ ไม่เพียงแต่ต้องมี**<u>รูปลักษณ์</u>**ที่น่าสนใจ แต่ต้อง**<u>ใช้งานได้</u>**เป็นอย่างดี ดังนั้นต้องเริ่มต้นด้วย “การวิเคราะห์ผู้ใช้” ซึ่งหมายถึงการทำความเข้าใจเกี่ยวกับผู้ใช้ระบบทั้งหมด 4 ด้านได้แก่

❑ **ทำความเข้าใจกับผู้ใช้งานระบบโดยตรง**

1. ผู้ใช้แต่ละกลุ่มมีทักษะและประสบการณ์การใช้งานคอมพิวเตอร์ในระดับใดบ้าง
2. ระดับการศึกษาโดยเฉลี่ยของผู้ใช้ทั้งหมด
3. ผู้ใช้สามารถเรียนรู้เนื้อหาการฝึกอบรมในชั้นเรียนได้มากน้อยเพียงใด
4. ผู้ใช้มีความชำนาญในการพิมพ์ดีดมากน้อยเพียงใด
5. อายุโดยเฉลี่ยของผู้ใช้ทั้งหมด
6. ผู้ใช้มีลักษณะการถูกครอบงำได้ง่ายหรือไม่
7. ค่าตอบแทนที่ผู้ใช้ได้รับจากการทำงานเท่าไร
8. ผู้ใช้ทำงานตรงเวลางาน หรือทำงานจนกว่างานจะแล้วเสร็จ
9. ซอฟต์แวร์ที่จะนำมาใช้เป็นส่วนหนึ่งของงานที่ต้องทำในแต่ละวันหรือบางโอกาส
10. ภาษาหลักที่ใช้สื่อสารกันในองค์กรคือภาษาใด
11. เมื่อใดมีความผิดพลาด มีวิธีการแก้ไขอย่างไร

## การวิเคราะห์ผู้ใช้ (User Analysis)

- **วิเคราะห์งานที่ผู้ใช้ต้องดำเนินการในแต่ละวัน**
  มีวัตถุประสงค์ของการวิเคราะห์งานที่ผู้ใช้ต้องดำเนินการในแต่ละวัน มีดังนี้
  1. เพื่อให้ทราบถึงลักษณะของการปฏิบัติงานในแต่ละสถานการณ์
  2. เพื่อให้ทราบถึงงานหลักและงานย่อยที่ต้องปฏิบัติ
  3. เพื่อให้ทราบถึงระบบงานและส่วนอื่นที่เกี่ยวข้องที่ผู้ใช้ต้องดำเนินการ
  4. เพื่อให้ทราบถึงลำดับของงาน (Workflow) ที่ต้องทำ
  5. เพื่อให้ทราบถึงลำดับของการปฏิบัติงาน (ลำดับการใช้ระบบในงานใดงานหนึ่ง)

- **วิเคราะห์ข้อมูลหรือสารสนเทศที่ต้องนำเสนอทางจอภาพหรือกระดาษ**
  ทีมงานต้องสอบถามผู้ใช้เพิ่มเติมเพื่อเก็บรายละเอียดของการนำเสนอสารสนเทศ โดยอาจตั้งคำถามต่อไปนี้
  1. ชนิดของข้อมูลหรือสารสนเทศที่จะต้องแสดงผลในแต่ละงานแตกต่างกันอย่างไร
  2. ผู้ใช้ต้องการจัดวางข้อมูลที่จะแสดงบนจอภาพคอมพิวเตอร์ด้วยตนเองหรือไม
  3. ผู้ใช้ต้องการให้จัด Layout ของเอกสารอย่างไร เพื่อให้เข้าใจได้ง่ายขึ้น





## การวิเคราะห์ผู้ใช้ (User Analysis)

- **วิเคราะห์ข้อมูลหรือสารสนเทศที่ต้องนำเสนอทางจอภาพหรือกระดาษ (ต่อ)**
  4. มีกลไกที่ใช้จัดทำ Pivot Table เพื่ออำนวยความสะดวกแก่ผู้ใช้ในส่วนของการสร้างรายงานด้วยตนเองหรือไม่
  5. จัดการแสดงผลแบบกราฟิกให้พอดีกับจอภาพได้ด้วยวิธีใด
  6. ควรเลือกใช้สีอย่างไรให้เหมาะสม
  7. ควรเลือกใช้ข้อความหรือถ้อยคำใดเพื่อแสดงเป็น Error Message

- **วิเคราะห์สภาพแวดล้อมในการทำงานของผู้ใช้**
  สิ่งหนึ่งที่ทีมงานไม่ควรมองข้าม คือ การตระหนักอยู่เสมอว่า การทำงานในองค์กรใด ๆ ไม่ว่าจะเป็นระบบใดก็ตาม "ผู้ใช้" ไม่ได้ทำงานเพียงคนเดียว แต่ทำงานอยู่ภายใต้สภาพแวดล้อมทางกายภาพอื่น ๆ ไม่ว่าจะเป็น อุปกรณ์ หรือความสัมพันธ์ระหว่างผู้ใช้ด้วยกันเอง หากซอฟต์แวร์ที่ผลิตขึ้นมาไม่สอดคล้องกับสภาพแวดล้อมอาจทำให้ผู้ใช้มีความรู้สึกสับสนในระหว่างใช้งานได้





## มาตรฐานการเขียนโปรแกรม

**เมื่อจบขั้นตอนการออกแบบ วิศวกรซอฟต์แวร์และทีมงานจะมีรายละเอียดในด้านต่าง ๆ ของซอฟต์แวร์ จากนั้นทีมงานก็จะทำรายละเอียดดังกล่าวมาสร้างซอฟต์แวร์ด้วย "การเขียนโปรแกรม (Program Writing)"**

สำหรับโปรแกรมเมอร์ การเขียนโปรแกรมไม่ใช่เรื่องยาก แต่สิ่งที่ยากกว่า คือ การที่โปรแกรมเมอร์ต้องคำนึงถึงหลักการต่าง ๆ ที่จะทำให้ผลิตภัณฑ์มีคุณภาพตามหลักของวิศวกรรมซอฟต์แวร์ เมื่อเริ่มเขียนโปรแกรม เหล่าโปรแกรมเมอร์จะร่วมกันสร้าง **ซอฟต์แวร์ในลักษณะชิ้นส่วน** แล้วจึงนำแต่ละชิ้นส่วนนั้นมาประกอบกัน ซึ่งต้องอาศัยโปรแกรมเมอร์จำนวนมาก ที่ใช้ทั้ง**เครื่องมือและเทคนิคต่าง ๆ มากมายร่วมกัน** นอกจากนั้นยังต้องคำนึงถึง**การทดสอบ** **การบำรุงรักษา**ภายหลังการใช้งานโปรแกรม และความสามารถใน**การนำมาใช้ใหม่**ได้ของโปรแกรม ซึ่งทั้งหมดนี้เพื่อให้ได้ซอฟต์แวร์ที่มีคุณภาพสูงตามหลักปฏิบัติของวิศวกรรมซอฟต์แวร์





## มาตรฐานการเขียนโปรแกรม

เพื่อให้การประกอบซอฟต์แวร์ การจัดทำเอกสาร และการบำรุงรักษาโปรแกรมจำนวนมากทำได้ง่าย จำเป็นต้องมีการกำหนดมาตรฐาน (Standard) และกระบวนการทำงาน (Procedure) ขึ้นมาเพื่อควบคุมการดำเนินงานของโปรแกรมเมอร์ ให้สอดคล้อง มีระเบียบ และเป็นไปในทิศทางเดียวกัน และก่อให้เกิดประโยชน์ดังนี้

- **ประโยชน์ต่อโปรแกรมเมอร์**
  ในแง่ของโปรแกรมเมอร์ มาตรฐานและกระบวนการ จะช่วยให้โปรแกรมเมอร์ จัดลำดับความคิดได้อย่างเป็นระบบขึ้น หลีกเลี่ยงข้อผิดพลาดในเบื้องต้นได้ เช่น การกำหนดชื่อตัวแปร ชื่อฐานข้อมูล ที่ไม่สอดคล้องกันทีมอื่น เป็นต้น ตามรูปแบบโครงสร้างโค้ด และรูปแบบเอกสารที่กำหนดไว้อย่างชัดเจน

## มาตรฐานการเขียนโปรแกรม

- **ประโยชน์ต่อบุคคลอื่น**
  โปรแกรมที่เขียนเสร็จเรียบร้อย จะต้องนำไปทดสอบ ประเมิน และประกอบโดยบุคคลอื่น ดังนั้นการทำโครงสร้าง รูปแบบ และเอกสารจะทำให้โปรแกรมเมอร์หรือทีมงานอื่นอ่านเข้าใจง่าย ว่าแต่ละส่วนของโปรแกรมมีหน้าที่อะไรและทำงานอย่างไร ซึ่งเราอาจเขียนในโครงสร้างโปรแกรมด้วย Comment พอสังเขป

เช่น
- โปรแกรมทำหน้าที่อะไร
- เรียกใช้อย่างไร
- มีตัวแปรอื่นใดที่เกี่ยวข้องบ้าง
- มีรายละเอียดสูตรคำนวณอะไร
- หรือเงื่อนไขอะไร เป็นต้น

```
/*=====================================*/
/*  Component to read input data from file          */
/*  Component Name : ListPersonFile                 */
/*  Programmer : Paijit Suksomboon                  */
/*  Version : 1.0 (1 Auguest 1996)                  */
/*                                                  */
/*  Function Invocation :                           */
/*      ListPerson(i);                              */
/*  Input Parameter :                               */
/*      i  from 1 until End of File                 */
/*  Struct Variable:                                */
/*     “Pers_name” from struct “Personnel”          */
/*=====================================*/
```





## หลักปฏิบัติในการเขียนโปรแกรม

เนื่องจากทุกโปรแกรมที่ถูกสร้างขึ้น ไม่ว่าจะใช้ภาษาใดก็ตาม จะต้องมีโครงสร้างหลักเหมือนกัน 3 อย่าง ได้แก่ **Control Structure, Algorithm** และ **Data Structure**

- **โครงสร้างควบคุมการทำงานของโปรแกรม (Control Structure)**
  หมายถึง การควบคุมการทำงานของส่วนประกอบย่อยของซอฟต์แวร์หลายรูปแบบ โดยโปรแกรมหลักมักเรียกโปรแกรมย่อยอาจเกิดขึ้นเมื่อระบบมีการเปลี่ยนสถานะหรือมีเหตุการณ์ (OO) หรือเขียนในลักษณะเพื่อให้กระทำ Procedural ซึ่งทุกรูปแบบล้วนมีความสำคัญ เพราะสะท้อนให้เห็นโครงสร้างการทำงานในระบบ อย่างไรก็ตาม เพื่อความสะดวกในการอ่านโค้ด และไม่ต้องกังวลกับทิศทางการทำงานของโปรแกรม จึงกำหนดแนวทางการเขียนโปรแกรมไว้ 2 แนวทาง คือ
  **(1) เขียนโค้ดจากบนลงล่าง**
  **(2) แบ่งส่วนการทำงานออกเป็นโมดูล**





## หลักปฏิบัติในการเขียนโปรแกรม

- **โครงสร้างควบคุมการทำงานของโปรแกรม (Control Structure)**
  **(1) เขียนโค้ดจากบนลงล่าง**
  คือ เขียนในลักษณะตามคำสั่งควบคุมของโปรแกรม กล่าวคือ ไม่ควรเขียน กระโดดไปมา เพราะทำให้อ่านโค้ดยุ่งยากแล้ว อาจทำให้การทำงานของโปรแกรมผิดพลาดได้

```
          dis = normal;
          if (price>=3000) goto A;
          dis = normal + bonus +1.5;
          dis = 0; goto C;
          if (price>=5000 goto B;
          if (price>=10000) goto C;
A :       if (price>=5000) goto B;
          dis = normal; goto C;
B :       if (price>=10000) goto C;
          dis = normal + bonus;
C :       next statement;
```

**Top-down**

```
If (price>=3000) dis = normal;
elseif (price>=5000) dis = normal +bonus;
elseif (price>=10000) dis = normal+bonus+1.5;
else dis=0;
```





## หลักปฏิบัติในการเขียนโปรแกรม

- **โครงสร้างควบคุมการทำงานของโปรแกรม (Control Structure)**
  **(2) แบ่งส่วนการทำงานออกเป็นโมดูล**
  การแบ่งส่วนการทำงานออกเป็นโมดูล แต่ละโมดูลรับผิดชอบการประมวลผล 1 หน้าที่ โดยโปรแกรมมิ่งแต่ละภาษาสร้างโมดูลแตกต่างกัน เช่น Macro, Subroutine, Procedure, Function, Method หรือ Inheritance เป็นต้น โดยจะมีการเรียกใช้เป็นระดับ ทำให้อ่านโค้ด ทดสอบ และบำรุงรักษาโปรแกรมได้ง่ายขึ้น

```
Main()
    Price = 5001;
    Dis=CalDiscount(Price);
    Tax=CalTax(Price);
    Total = Price-Dis+Tax;
End;
```

```
CalDiscount(X)
/*ส่วนลด 10% สำหรับยอดซื้อ(x) 5000 บาทขึ้นไป    */
    if X>=5000
        CalDiscount=Price*0.1
    else  CalDiscount=0;
```

```
CalTax(Y)
/*ภาษี ณ ที่จ่าย 10% สำหรับยอดซื้อ(y) > 5000 บาท   */
/*ภาษี ณ ที่จ่าย 7% สำหรับยอดซื้อ(y)ไม่เกิน 5000 บาท*/
    if Price>5000
        CalTax=Price*0.10
    else  CalTax=Price*0.07;
```

## หลักปฏิบัติในการเขียนโปรแกรม

- **ขั้นตอนวิธี/อัลกอริทึม (Algorithm)**
  ไม่ว่าจะออกแบบซอฟต์แวร์ด้วยแนวทางเชิงโครงสร้างหรือแนวทางเชิงวัตถุ จะพบว่า ทีมงานออกแบบจะกำหนดอัลกอริทึมเพื่อใช้ประมวลผลข้อมูลในแต่ละส่วนประกอบย่อยของซอฟต์แวร์มากให้ด้วย
  ***อัลกอริทึม*** เป็นลำดับขั้นตอนการทำงานหรือแก้ปัญหางานใด ๆ โดยการอธิบายด้วยเครื่องมือหลายชนิด เช่น บรรยาย/ภาษาอังกฤษเชิงโครงสร้าง () ผังงานโปรแกรม (Flowchart) หรือรหัสเทียม (Psudocodes) เป็นต้น

  **หน้าที่ของโปรแกรมเมอร์**คือ นำอัลกอริทึมที่ได้จากการออกแบบมาเขียนเป็นโค้ดโปรแกรมภาษาใต้ภาษาโปรแกรมมิ่ง แพลตฟอร์ม และฮาร์ดแวร์ที่กำหนด โดยมีหลักที่ต้องคำนึงคือ "**ต้องเขียนโค้ดที่มีประสิทธิภาพและประสิทธิผล**" กล่าวคือ

> **การทำงานรวดเร็ว ผลลัพธ์ที่ได้ถูกต้องแม่นยำ ตรงตามมาตรฐาน และความต้องการของผู้ใช้**





## หลักปฏิบัติในการเขียนโปรแกรม

- **โครงสร้างข้อมูล (Data Structure)**
  ในที่นี้หมายถึง ลักษณะของข้อมูลที่จะถูกประมวลผลในโปรแกรม เพราะบ่อยครั้งมักพบว่า ข้อมูลที่รับเข้ามาเป็นตัวกำหนดโครงสร้างการทำงานของโปรแกรม ดังนั้น จึงมีข้อแนะนำในการเขียนโปรแกรมตามโครงสร้างข้อมูล ดังนี้

  **(1) เขียนโปรแกรมให้เรียบง่ายที่สุด**
  กล่าวคือ ไม่ว่าขั้นตอนการออกแบบจะกำหนดข้อมูลมาอย่างไร ให้นำมาปรับใหม่เพื่อทำให้สามารถเขียนโปรแกรมได้ง่ายขึ้น

| ขั้นเงินได้สุทธิตั้งแต่ | อัตราภาษี (ร้อยละ) |
|---|---|
| 0-100,000 | ยกเว้น |
| 100,001-500,000 | 10 |
| 500,001-1,000,000 | 20 |
| 1,000,001-4,000,000 | 30 |
| 4,000,001 บาทขึ้นไป | 37 |

**ตัวอย่าง**
1. นาย ก. เงินได้สุทธิ 90,000 บาท
2. นาย ข. เงินได้สุทธิ 300,000 บาท
3. นาย ค. เงินได้สุทธิ 650,000 บาท
4. นาย ง. เงินได้สุทธิ 1,500,000 บาท





## หลักปฏิบัติในการเขียนโปรแกรม

- **โครงสร้างข้อมูล (Data Structure)**
  **(1) เขียนโปรแกรมให้เรียบง่ายที่สุด (ต่อ)**
  เพื่อให้เขียนโปรแกรมให้ง่ายขึ้น สามารถจัดโครงสร้างข้อมูลในตารางใหม่ได้

| ขั้นเงินได้สุทธิตั้งแต่ | เงินได้สุทธิสูงสุดของขั้น | อัตราภาษี (ร้อยละ) | ภาษีในแต่ละขั้นเงินได้ | ภาษีสะสมสูงสุดของขั้น |
|---|---|---|---|---|
| 0-100,000 | 100,000 | ยกเว้น | ยกเว้น | 0 |
| 100,001-500,000 | 400,000 | 10 | 40,000 | 40,000 |
| 500,001-1,000,000 | 500,000 | 20 | 100,000 | 140,000 |
| 1,000,001-4,000,000 | 3,000,000 | 30 | 900,000 | 1,040,000 |
| 4,000,001 บาทขึ้นไป | | 37 | | |

**ตัวอย่าง**
1. นาย ก. เงินได้สุทธิ 90,000 บาท
2. นาย ข. เงินได้สุทธิ 300,000 บาท
3. นาย ค. เงินได้สุทธิ 650,000 บาท
4. นาย ง. เงินได้สุทธิ 1,500,000 บาท





## หลักปฏิบัติในการเขียนโปรแกรม

- **โครงสร้างข้อมูล (Data Structure)**
  **(1) เขียนโปรแกรมให้เรียบง่ายที่สุด (ต่อ)**
  และนำมาเขียนอัลกอริทึมงานคำนวณภาษีเงินได้บุคคลธรรมดา

```
Tax := 0.00;
IF (income<= 100000) THEN Tax := 0;
ELSE IF (income <= 500000)      THEN Tax := ((income-100000)*0.10);
ELSE IF (income <= 1000000)     THEN Tax := ((income-500000)*0.20)+40000;
ELSE IF (income <= 4000000)     THEN Tax := ((income-1000000)*0.30)+100000;
ELSE Tax := ((income-4000000)*0.37)+900000;
```

  **(2) ใช้โครงสร้างข้อมูลเป็นตัวกำหนดโครงสร้างโปรแกรม**
  จากหลักปฏิบัติในข้อ 1 เป็นตัวกำหนดโครงสร้างใหม่ เพื่อให้การเขียนโปรแกรมเรียบง่ายขึ้น โดยพิจารณาให้รอบคอบว่าภาษาโปรแกรมมิ่งใดที่เหมาะสมกับโครงสร้างข้อมูลที่มีอยู่ส่วนใหญ่

## Implementation Phase

### หลักปฏิบัติอื่น ๆ

นอกจากหลักปฏิบัติที่เกี่ยวข้องกับองค์ประกอบทั้ง 3 ส่วนแล้ว ยังมีหลักปฏิบัติอื่น ๆ ที่ช่วยให้การเขียนโปรแกรมมีคุณภาพ ดังนี้

(1) แยกฟังก์ชันหรือโมดูลรับและแสดงผลข้อมูลออกมาต่างหากจากโค้ดส่วนอื่นของโปรแกรม ทั้งนี้ เพื่อให้การแก้ไขหรือปรับปรุงในส่วนดังกล่าวง่ายขึ้น และยังลดจำนวนการเขียนโค้ดซ้ำ ๆ ลงได้ อีกทั้งยังอ่านโค้ดเข้าใจง่ายขึ้นอีกด้วย

(1) ใช้วิธีการเขียนรหัสเทียมเพื่อทดลองออกแบบอัลกอริทึมของโปรแกรมก่อนลงมือเขียนโค้ดจริง เพราะสามารถใช้แก้ไขปัญหาใด ๆ อย่างเป็นลำดับขั้นตอน และสามารถปรับรหัสเทียมให้เกิดการกระชับที่สุดได้ เมื่อนำมาเขียนโค้ดจริงย่อยทำให้โค้ดนั้นมีประสิทธิภาพกว่า ผิดพลาดน้อยกว่า และมีผลทำให้แก้ไขโค้ดนั้นน้อยลงด้วย



## Implementation Phase

### หลักปฏิบัติอื่น ๆ

นอกจากหลักปฏิบัติที่เกี่ยวข้องกับองค์ประกอบทั้ง 3 ส่วนแล้ว ยังมีหลักปฏิบัติอื่น ๆ ที่ช่วยให้การเขียนโปรแกรมมีคุณภาพ ดังนี้

(3) โดยทั่วไปเมื่อเริ่มต้นเขียน โปรแกรมเมอร์ควรจะใช้วิธีเขียนโค้ดให้ทำงานได้คร่าว ๆ แล้วจึงมาเก็บรายละเอียดเพิ่มเติมภายหลังจากทดสอบผลลัพธ์แล้วพบว่าถูกต้อง เพราะหากผลลัพธ์มีข้อบกพร่องหรือผิดพลาดเราสามารถย้อนกลับมาพิจารณาการออกแบบและโครงสร้างได้ ทำให้แก้ปัญหาได้ตรงจุดมากกว่า

(4) สำหรับประเด็นการนำมาใช้ซ้ำ (Reuse) ควรพิจารณาตามผู้ที่เกี่ยวข้อง 2 ฝ่าย คือ ผู้ผลิตคอมโพเนนท์สามารถใช้ซ้ำได้ (Producer Reuse) และลูกค้านำคอมโพเนนท์มาใช้ซ้ำ (Consumer Reuse) โดยต้องหลักการผลิตที่ต้องคำนึงถึงแตกต่างกัน



## Implementation Phase

### กระบวนการเขียนโปรแกรม

โปรแกรมที่มีคุณภาพ ไม่ได้มาจากการออกแบบที่มีคุณภาพเพียงปัจจัยเดียว แต่ยังประกอบไปด้วยปัจจัยอื่น ๆ ได้แก่ ทักษะ ประสบการณ์ และ ความเฉลียวฉลาดที่เกิดจากการมีจินตนาการและกระบวนการแก้ปัญหาที่ดีของโปรแกรมเมอร์ ดังนั้นกระบวนการเขียนโปรแกรมจึงเชื่อมโยงกับกระบวนการแก้ปัญหา ซึ่งต้องทำความเข้าใจแบ่งเป็น 4 ขั้นตอน

(1) ทำความเข้าใจปัญหา
แบ่งปัญหาออกเป็นส่วนย่อย แล้ววิเคราะห์แต่ละส่วนให้เกิดความเข้าใจ ซึ่งสำหรับโปรแกรมเมอร์ควรใช้ Flowchart เป็นเครื่องมือในการจำลองลำดับขั้นตอนการทำงาน

(2) วางแผนแก้ไขปัญหา
การออกแบบวิธีแก้ไขปัญหาที่ผ่านการวิเคราะห์มาแล้ว โดยพิจารณาหาส่วนที่เกี่ยวข้องกับปัญหาเช่น ข้อมูล อัลกอริทึม ไลบรารี หรือระเบียบปฏิบัติ เพื่อนำมาปรับใช้

(3) ดำเนินการตามแผน
เมื่อวางแผนการแก้ไขปัญหาเรียบร้อยแล้ว ขั้นต่อมาคือดำเนินการตามแผน ในที่นี้ คือ "ลงมือเขียนโค้ดโปรแกรม"

(4) ทบทวน
ย้อนกลับไปพิจารณาสิ่งที่ดำเนินการแก้ไขแต่ละส่วนและประเมินว่าเหมาะสมหรือถูกต้องที่สุดแล้วหรือไม่ หรือดีที่สุดแล้วสามารถนำมาเป็น Pattern ใช้ซ้ำได้อีก