8 พอยท์เตอร์ (Pointers)

ทุกครั้งที่เราเรียกฟังก์ชั่นที่รับอาร์กิวเมนท์ อาร์กิวเมนท์เหล่านั้นจะถูกส่งเข้าไปในรูปแบบของการคัดลอกดัง ตัวอย่างเช่น

```
func zero(x int) {
      x=0
}
func main() {
      x := 5
      zero(x)
      fmt.Println(x) // x is still 5
}
```

โปรแกรมด้านบนเราจะเห็นว่าฟังก์ชั่น zero จะไม่ทำการแก้ไขค่าเริ่มต้นของตัวแปร x จากฟังก์ชั่น main แต่จะทำยังไงถ้าเราต้องการเปลี่ยนค่า หนึ่งในทางออกสำหรับความต้องการนี้คือการใช้ประเภทของข้อมูลชนิดพิเศษที่เรียกว่าพอยท์เตอร์(pointer)

```
func zero(xPtr *int) {
      *xPtr = 0
}
func main() {
      x := 5
      zero(&x)
      fmt.Println(x) // x is 0
}
```

พอยท์เตอร์จะอ้างถึงตำแหน่งในหน่วยความจำที่ค่าถูกจัดเก็บไว้แทนที่จะเป็นการอ้างถึงค่าที่ถูกจัดเก็บไว้ (ชี้ไปอย่างอื่น) ดังนั้นการใช้พอยท์เตอร์ (*int) จะทำให้ฟังก์ชั่น zero สามารถเปลี่ยนค่าเริ่มต้นของตัวแปรได้

8.1 เครื่องหมาย * และ &

พอยท์เตอร์ในภาษาโกจะใช้เครื่องหมาย * (ดอกจัน) และตามด้วยประเภทของตัวแปรที่ถูกจัดเก็บ ดังนั้นในฟังก์ชั่น zero เราเขียน *xPtr เป็นการบอกว่า xPtr ชี้ไปที่ int นอกจากนี้การใช้ * ยังใช้สำหรับ “dereference” ตัวแปรของพอยท์เตอร์ด้วย การทำ dereference หมายถึงการเข้าถึงค่า(value)ที่พอยท์เตอร์นั้นชี้(point)อยู่ยกตัวอย่างเช่นถ้าเราเขียน *xPtr = 0 เรากำลังทำ “เก็บค่า 0 ไว้ที่หน่วยความ

จำที่ xPtr อ้างถึงอยู่" แต่จะเกิดอะไรขึ้นถ้าเราเขียน xPtr = 0 สิ่งที่เราจะได้คือข้อผิดพลาดจากการคอมไพล์เพราะ xPtr ไม่ใช่ int แต่เป็น *int ดังนั้นสิ่งที่เราทำได้คือการส่งค่า *int (พอยท์เตอร์ไปที่ int) ให้มันเท่านั้นซึ่งสิ่งที่เราทำคือการส่งตำแหน่งของ x เข้าไปแทน นั่นเป็นสิ่งที่ทำให้เราสามารถแก้ไขค่าของตัวแปร &x ในฟังก์ชั่น main นั่นก็เพราะ xPtr ในฟังก์ชั่น zero อ้างไปที่ตำแหน่งของหน่วยความจำตำแหน่งเดียวกัน

8.2 new

อีกทางที่เราสามารถใช้ pointer ได้คือการใช้ฟังก์ชั่น new ที่มีมาให้แล้วใน Go:

```
func one(xPtr *int) {
     *xPtr = 1
}
func main() {
     xPtr := new(int)
     one(xPtr)
     fmt.Println(*xPtr) // x is 1
}
```

เราจะเห็นว่า new รับไทป์(type) เป็นอากิวเมนท์และสิ่งที่มันทำคือการจอง(allocate)หน่วยความจำให้พอดีกับค่าของข้อมูลประเภทนั้นๆและจากนั้นก็ทำการส่งพอยท์เตอร์กลับออกมา มีเรื่องน่าสนใจเกี่ยวกับการใช้ new และ & ในภาษาอื่นๆเพราะในภาษาเหล่านั้นการใช้ new และ & มีความแตกต่างกันอย่างมากการใช้งานต้องทำด้วยความระมัดระวังไม่เช่นนั้นแล้วสิ่งที่เราสร้างไว้ด้วย new จะต้องถูกลบ แต่สำหรับโก สิ่งนี้จะไม่เกิดขึ้นเพราะโกเป็นภาษาที่ใช้ตัวจัดการขยะ(Gabage Collector) นั่นหมายความว่าหน่วยความจำจะถูกเก็บกวาดแบบอัตโนมัติเมื่อไม่มีการใช้งานมันอีกต่อไป แต่สำหรับGo การใช้งานพอยท์เตอร์กับ built-in type เป็นเรื่องที่เกิดขึ้นไม่บ่อยเท่าไหร่นักแต่อย่างไรก็ตามในบทต่อไปเราจะได้เห็นว่าพอยท์เตอร์จะมีประโยชน์สูงมากเมื่อใช้งานคู่กับสตรัคท์