บทที่ 10

ชื่อและชนิดข้อมูล

10.1 ชื่อและคุณสมบัติ

10.2 ตัวแปร

10.3 ชนิดข้อมูล

10.4 เอกสารอ้างอิงและเว็บไซต์ที่ควรรู้

วัตถุประสงค์

- ชื่อและการเชื่อมโยงชื่อเข้ากับคุณสมบัติต่างๆ
- ตัวแปรและคุณสมบัติที่เกี่ยวข้อง
- ชนิดข้อมูลประเภทต่างๆ

ชื่อของตัวแปร ฟังก์ชัน ชนิดข้อมูล คลาส และสิ่งอื่นๆ ที่ใช้ในการเขียนโปรแกรม จะต้องมีการเชื่อมโยงเข้ากับคุณสมบัติของชื่อเหล่านั้นเสมอ เรียกว่าการ binding การศึกษาให้เข้าใจถึงการนำชื่อของสิ่งต่างๆ เหล่านี้ไปใช้งาน การอ้างถึงชื่อในบริบทที่แตกต่างกัน เป็นทักษะสำคัญของการเขียนโปรแกรม ในบทนี้เราจะกล่าวถึง syntax พื้นฐานของชื่อ คำพิเศษในภาษาโปรแกรม คุณสมบัติของตัวแปร ซึ่งประกอบด้วยชนิดข้อมูล แอดเดรส และค่าของตัวแปร นอกจากนี้ยังกล่าวถึงหลักการ binding และ binding time รวมทั้งชนิดของข้อมูล ทั้งที่เป็นชนิดข้อมูลพื้นฐาน และชนิดข้อมูลแบบมีโครงสร้าง

10.1 ชื่อ (Names)

ก่อนที่จะพูดถึงตัวแปร จะขอพูดถึงคุณสมบัติพื้นฐานอย่างหนึ่งของตัวแปรก่อนนั่นก็คือ ชื่อ นอกจากชื่อจะเกี่ยวข้องกับตัวแปรแล้ว ยังเกี่ยวกับโครงสร้างอื่นๆ ในโปรแกรมด้วย เช่น ค่าคงที่ โปรแกรมย่อย พารามิเตอร์ ชื่อหรือเรียกอีกอย่างหนึ่งว่า Identifier หมายถึงสตริงที่ใช้ในการระบุสิ่งต่างๆ (entity) ในโปรแกรม สิ่งที่ต้องพิจารณาในการตั้งชื่อก็ขึ้นอยู่กับ syntaxk ของภาษาโปรแกรม ซึ่งมักจะมีกฎเกณฑ์ที่เกี่ยวข้องดังต่อไปนี้

- ความยาวของชื่อ (Maximum length) ภาษาโปรแกรมในยุคแรกมักกำหนดให้ชื่อมีความยาวไม่มากนัก เช่น Fortran กำหนดให้มีความยาว 6 ตัวอักษรและเป็นตัวใหญ่ทั้งหมด Fortran ในยุคหลังจึงเริ่มขยายความยาวขึ้น เช่น Fortran 95 กำหนดให้ไม่เกิน 31 ตัวอักษร ในบางภาษาไม่มีการจำกัดความยาวของชื่อ แต่อาจจะให้ความสำคัญเฉพาะอักษรเริ่มต้นจำนวนหนึ่งเช่น ภาษา C บางเวอร์ชั่นหรือในบางภาษาเช่น C# และ Java ก็ไม่จำกัดความยาวและให้ความสำคัญกับทุกตัว
- ตัวอักขระพิเศษ (Special character) หลายๆ ภาษาอนุญาตให้ใช้ตัวอักขระพิเศษได้ บางภาษาอาจให้ใช้เพียงตัวเดียวหรือบางตัวเท่านั้น เช่น ภาษาในกลุ่มเดียวกับภาษา C อนุญาตให้ใช้ ขีดล่าง (_ underscore) ได้เท่านั้น แต่ไม่มีข้อกำหนดในการใช้ บางภาษา เช่น Cobol อนุญาตให้ใช้ ขีด (- hyphen) ได้แต่ต้องไม่ใช้ขึ้นต้นหรือลงท้ายชื่อ
- ความแตกต่างระหว่างตัวอักษรเล็กและตัวอักษรใหญ่ (Case sensitive) ภาษาในกลุ่ม Pascal หรือ Ada จะไม่คำนึงถึงตัวอักษรเล็กใหญ่ ดังนั้นการเขียนชื่อด้วยตัวอักษรเล็กใหญ่ต่างกัน เช่น alpha, Alpha หรือ alpha ก็ถือเป็นชื่อเดียวกัน ซึ่งจะตรงข้างกับภาษาในกลุ่มภาษา C จะถือว่าชื่อทั้งสามไม่ใช่ชื่อเดียวกัน นอกจากนี้ยังมีบางภาษาที่สร้างความสับสนได้เช่นกัน เช่น PHP ที่ออกแบบให้มีทั้งสองกรณีขึ้นอยู่กับลักษณะการนำไปใช้
- คำพิเศษ (Special word) ในภาษาโปรแกรมโดยมากจะมีกลุ่มชื่อที่กำหนดไว้ก่อนแล้ว เพื่อนำไปใช้หรือมีความหมายเฉพาะอย่าง และไม่สามารถนำมาใช้ในการตั้งชื่อได้ กลุ่มคำเหล่านี้เรียกว่า คำสงวน (Reserved word) หรือ คำสำคัญ (Keyword) ข้อแตกต่างคือ Keyword จะขึ้นอยู่กับบริบทรอบข้าง อาจนำมาใช้ตั้งชื่อได้ แต่ Reserved word จะไม่สามารถนำมาใช้ตั้งชื่อได้ ตัวอย่างเช่น ในภาษา Fortran คำว่า `Real` เมื่อขึ้นต้นประโยคแล้วตามด้วยชื่อจะถือเป็น keyword ซึ่งมีความหมายว่าเป็นการประกาศตัวแปร แต่หากตามด้วยเครื่องหมายกำหนดค่า จะถือเป็นชื่อตัวแปร ดังตัวอย่าง

```
Real Apple
Real = 3.4
```

10.2 ตัวแปร (Variables)

ตัวแปรเป็นสิ่งที่ใช้ในการอ้างถึงเซลล์หรือกลุ่มเซลล์ในหน่วยความจำหลัก ซึ่งถือเป็นสิ่งสำคัญในการเขียนโปรแกรมเชิงคำสั่งและเชิงวัตถุ ในภาษาทั้งสองแบบ ตัวแปรก็คือ ชื่อที่ใช้อ้างถึงตำแหน่งของหน่วยความจำ (Memory location) นั่นคือตัวแปรจะเป็นการ binding ชื่อกับตำแหน่ง (address) ชนิดข้อมูล (type) ค่า (value) และอายุการใช้งาน (lifetime) นอกจากนี้ ขอบเขตของตัวแปรก็เป็นอีกคุณสมบัติหนึ่งที่สำคัญที่จำเป็นต้องรู้

การ binding นั้นมี 2 ลักษณะ ได้แก่

- static ที่การ binding จะเกิดขึ้นก่อนรันโปรแกรม (before run-time)
- dynamic ที่การ binding จะเกิดขึ้นขณะรันโปรแกรม (at run-time)

เวลาในที่เกิดการ binding สำหรับชื่อเป็นเรื่องสำคัญ ตัวอย่างเช่น ภาษาโดยมากมักจะใช้ static scoping คือ ต้องมีการประกาศชื่อก่อนการเรียกใช้เสมอ แต่ก็มีบางภาษาที่ยังไม่จำเป็นต้องประกาศจนกว่าจะมีการใช้งาน การ binding ของชื่อตัวแปรกับคุณสมบัติต่างๆ นั้นอาจเป็นได้ทั้งแบบ static และ dynamic

ตำแหน่งในหน่วยความจำ (Address)

ตัวแปรจะต้องมีการเก็บค่าไว้ในหน่วยความจำ ซึ่งจะต้องมีการระบุตำแหน่ง (address) ของตัวแปร ซึ่งตำแหน่งของตัวแปรในภาษาโปรแกรมโดยส่วนมาก จะมีลักษณะดังต่อไปนี้

- ตัวแปรชื่อเดียวกันอาจมีตำแหน่งที่ต่างกัน เมื่อปรากฏอยู่คนละที่ และเรียกใช้ในเวลาที่ต่างกันภายในโปรแกรมเดียวกัน ตัวอย่างเช่น โปรแกรมประกอบด้วยโปรแกรมย่อยชื่อ sub1 และ sub2 ซึ่งทั้งสองต่างก็มีตัวแปรแบบ local ชื่อ sum เหมือนกัน ดังนั้น ตัวแปรทั้งสองต่างก็ไม่เกี่ยวข้องกัน
- ตัวแปรเดียวกันอาจมีตำแหน่งที่ต่างกัน ในเวลาที่ต่างกันขณะทำการรันโปรแกรม ตัวอย่างเช่น ถ้าโปรแกรมย่อยมีตัวแปรแบบ local เมื่อมีการเรียกใช้โปรแกรมย่อยแต่ละครั้ง การระบุตำแหน่งของตัวแปรก็อาจจะแตกต่างกัน
- ถ้าตัวแปรสองตัวสามารถเข้าถึงตำแหน่งในหน่วยความจำตำแหน่งเดียวกันได้ จะเรียกว่า Alias ตัวอย่างเช่น ตัวแปร total และ sum เป็น alias ถ้ามีการเปลี่ยนค่าของตัวแปร total ก็จะมีผลกับค่าของตัวแปร sum ด้วยเช่นกัน

ค่าของตัวแปร (Value)

ค่าของตัวแปร หมายถึง ค่าที่เก็บอยู่ในหน่วยความจำ ซึ่งอาจเป็นจำนวนเต็ม จำนวนจริง ตัวอักษร ตำแหน่งในหน่วยความจำ หรือข้อมูลชนิดอื่นๆ ขึ้นอยู่กับการกำหนดชนิดของตัวแปร ซึ่งบางครั้งเมื่อเราอ้างถึงตัวแปรอาจจะเกิดความสับสนว่าต้องการอ้างถึงข้อมูลที่เก็บอยู่ในตัวแปร หรืออ้างถึงตำแหน่งของตัวแปรนั้น

ภาษา Algol 68 เป็นภาษาแรกที่ระบุความแตกต่างของการใช้ชื่อของตัวแปรแทนตำแหน่ง (l-value) กับการใช้ชื่อของตัวแปรแทนค่าที่เก็บอยู่ (r-value) พิจารณาคำสั่งกำหนดค่าต่อไปนี้

```
x = y + 1;
```

จากคำสั่งข้างต้น เป็นการกำหนดค่าให้กับตำแหน่งในหน่วยความจำที่แทนด้วยตัวแปร x ซึ่งหาได้จากผลบวกของค่าของตัวแปร y บวกกับ 1 สังเกตว่า ตัวแปร x ที่อยู่ทางซ้ายมือ (l-value) หมายถึงตำแหน่ง ขณะที่ตัวแปร y ที่อยู่ด้านขวามือ (r-value) หมายถึงค่าที่เก็บอยู่ในตำแหน่ง

ในบางภาษาเช่น ภาษา ML จะสนับสนุนการอ้างตัวแปรแบบ explicit dereferencing คือใช้ระบุการอ้างตัวแปรว่าเป็นการอ้างถึงค่าหรือแอดเดรสไว้อย่างชัดเจน เช่น การใช้ operator ! ในการอ้างถึงค่าที่เก็บใน y

```
x = !y + 1;
```

สำหรับตัวแปรแบบ pointer การใช้ explicit dereferencing ก็มีประโยชน์ เช่น ในภาษา C/C++ จะใช้เครื่องหมาย * ในการอ้างถึงตัวแปรแบบ pointer ดังตัวอย่าง

```
1    int x=1, y=2;
2    int* p;
3    p = &x;
4    y = *p;
```

จากตัวอย่าง คำสั่งบรรทัดแรกเป็นการประกาศตัวแปร x และ y ทำให้เกิดการจองพื้นที่สำหรับตัวแปรทั้งสอง แล้วนำค่า 1 และ 2 ไปใส่ในตำแหน่งนั้นตามลำดับ บรรทัดที่สองเป็นการประกาศตัวแปรแบบ pointer ชื่อ p ที่ใช้เก็บค่า int บรรทัดที่สาม นำตำแหน่งของตัวแปร x ไปใส่ยังตำแหน่งของ p นั่นคือกำหนดให้ pointer p ชี้ไปที่ตำแหน่งของ x และบรรทัดสุดท้ายเป็นกำหนดค่าให้กับตัวแปร y โดยนำค่าที่ได้จากการ dereference p ซึ่งก็คือค่าที่เก็บอยู่ในตำแหน่งของ x นั่นเอง ผลการรันจะได้ค่าของตัวแปร x, y และ *p มีค่าเท่ากับ 1

ตัวแปรอาจแบ่งเป็น 2 ประเภทคือ

- static variable ตัวแปรชนิดนี้จะรู้ขนาดโครงสร้างที่แน่นอน เมื่อมีการกำหนดตัวแปรขึ้น การ binding เกิดขึ้นก่อนช่วงเวลา run-time เช่น การประกาศตัวแปร int x ในภาษา C จะทำการจองเนื้อที่ในหน่วยความจำหลักไว้สำหรับเก็บข้อมูลประเภทจำนวนเต็ม และทำการ binding กับชื่อ x เมื่อมีการอ้างถึงตัวแปร x ก็จะเรียกใช้ค่าที่เก็บอยู่ในตำแหน่งนี้
- dynamic variable ตัวแปรชนิดนี้จะมีการขอเนื้อที่หน่วยความจำเมื่อต้องการใช้งานเท่านั้น ดังนั้น การ binding เกิดขึ้นในช่วงเวลา run-time ค่าที่บรรจุในตัวแปรชนิดนี้มักจะเป็นตำแหน่งของหน่วยความจำซึ่งเก็บข้อมูลที่ต้องการ ตัวอย่างเช่น ตัวแปรประเภท pointer หรืออาจเป็นการประกาศตัวแปรแบบอาร์เรย์โดยไม่ระบุขนาด

ชนิดข้อมูล (Type)

ชนิดของตัวแปรจะเป็นการกำหนด ช่วงค่าของตัวแปร (range of values) ที่เป็นไปได้ และกลุ่มของตัวดำเนินการ (set of operations) ที่กำหนดให้กระทำกับค่าของชนิดข้อมูลนั้นๆ ตัวอย่างเช่น ชนิดข้อมูล int ในภาษา Java จะมีค่าอยู่ในช่วง -2147483648 ถึง 2147483647 และมีตัวดำเนินการทางคณิตศาสตร์ที่ทำการบวก (+) ลบ (-) คูณ (*) หาร (/) และหารเอาเศษ (%) กับค่าเหล่านี้ได้ สำหรับชนิดข้อมูลจะกล่าวถึงโดยละเอียดในหัวข้อ 10.3

อายุการใช้งาน (Lifetime)

อายุการใช้งานของตัวแปร หมายถึง ช่วงเวลาที่มีการกำหนดพื้นที่ในหน่วยความจำให้กับตัวแปร ซึ่งสามารถแบ่งออกเป็น

- static – ทำการ binding ก่อนจะเริ่ม execution (load time binding)
- stack-dynamic – ทำการ binding เมื่อมีการประกาศตัวแปรตัวนั้น (runtime binding)
- explicit heap-dynamic – มีการกำหนดพื้นที่/เซลล์ในหน่วยความจำ (allocation) หรือคืนพื้นที่ในหน่วยความจำ (deallocation) โดยใช้คำสั่งโดยตรง
- implicit heap-dynamic - มีการ allocation หรือ deallocation โดยใช้คำสั่งกำหนดค่า

ขอบเขต (Scope)

คอมพิวเตอร์ในยุคแรกมีหน่วยความจำที่จำกัด การเขียนโปรแกรมจึงต้องคำนึงถึงการใช้พื้นที่ในหน่วยความจำด้วย จึงทำให้เกิดแนวคิดในการกำหนดขอบเขตของชื่อหรือตัวแปร ขอบเขตของชื่อ หมายถึง บริเวณหรือส่วนของโปรแกรมที่สามารถเรียกใช้หรือเข้าถึงชื่อได้ หรือเรียกว่าโปรแกรมสามารถมองเห็น (visible) ได้เฉพาะชื่อที่อยู่ในขอบเขตที่กำหนด

สำหรับ static scoping จะทำการ binding ชื่อกับส่วนของโปรแกรมตามตำแหน่งที่ปรากฏใน source program ดังนั้น static scoping จะสามารถตรวจสอบได้ในช่วงเวลาคอมไพล์ เนื่องจากการ binding ชื่อสามารถทำได้โดยการพิจารณาจากตัวโปรแกรม นอกจากนี้ยังช่วยทำให้โปรแกรมอ่านได้ง่าย และสนับสนุนการตรวจ สอบโปรแกรมในขณะคอมไพล์ด้วย ภาษาสมัยใหม่ เช่น C/C++, Java, Python ต่างก็ใช้ static scoping ทั้งสิ้น ขอบเขตของตัวแปร คือ ส่วนของโปรแกรมที่สามารถมองเห็นตัวแปรที่กำหนดและเรียกใช้งานได้ เช่น ในภาษา C หรือ Pascal ตัวแปรแบบ global จะสามารถเรียกใช้งานได้จากจุดที่มีการประกาศไว้จนจบโปรแกรม ส่วนตัวแปรแบบ local จะถูกจำกัดให้ใช้ได้เฉพาะ block ที่มีการประกาศตัวแปรไว้เท่านั้น ดังตัวอย่างโปรแกรมภาษา C ต่อไปนี้

```
1     void sort (float a[ ], int size) {
2           int i,j;
3           for (i=0; i<size; i++)
4               for (j=i; j<size; j++)
5                   if (a[j] < a[i] {
6                       float t;
7                       t = a[i];
8                       a[i] = a[j];
9                       a[j] = t;
10                  }
11    }
```

จากตัวอย่าง function sort ข้างต้น ขอบเขตของตัวแปร t จะอยู่ในช่วงที่ประกาศไว้คือบรรทัดที่ 6-9 และไม่มี block ที่ซ้อนอยู่ในช่วงนี้ ดังนั้น t จึงเป็นการอ้างถึงแบบ local และสามารถอ้างถึงได้เฉพาะคำสั่งที่อยู่ภายในเครื่องหมาย { ในบรรทัดที่ 5 และ } ในบรรทัดที่ 10 เท่านั้น นอกจากนี้ภาษา C กำหนดให้ต้องมีการ

ประกาศตัวแปรก่อนมีการเรียกใช้ในคำสั่งด้วย ส่วนในภาษา C++ และ Java อนุญาตให้มีการอ้างถึงชื่อก่อนที่จะมีการประกาศได้ เรียกว่า forward reference จากตัวอย่างเดิม ขอบเขตของตัวแปร a และ size จะอยู่ในช่วงบรรทัดที่ 2-10 นั่นคืออยู่ภายในเครื่องหมาย { ในบรรทัดที่ 1 และ } ในบรรทัดที่ 11 เท่านั้น ขณะที่ขอบเขตของตัวแปร i และ j จะอยู่ภายใน block เดียวกัน แต่จะอ้างถึงจะต้องเกิดขึ้นภายหลังจากการประกาศตัวแปรในบรรทัดที่ 2 โดยการอ้างถึงตัวแปร i ในช่วงบรรทัดที่ 3-5 และอ้างถึงตัวแปร j ในช่วงบรรทัดที่ 4-5 เป็นแบบ local ในขณะที่การอ้างถึงตัวแปรทั้ง i และ j ในบรรทัดที่ 7-9 เป็นแบบ nonlocal เนื่องจากบรรทัดที่ 6-10 เป็นขอบเขตย่อยที่ซ้อนอยู่ภายในขอบเขตที่มีการประกาศตัวแปร i และ j

ขอบเขตที่น่าสนใจอีกรูปแบบหนึ่งคือคำสั่ง for ใน C++ หรือ Java ที่มีอาจมีการประกาศตัวแปรควบคุมใหม่ ในกรณีนี้ขอบเขตของตัวแปรควบคุมนี้จะจำกัดอยู่เฉพาะภายในลูปเท่านั้น ดังตัวอย่าง

```
for (int i=0; i<10; i++) {
      System.out.println(i);
      …
}
... i ... // invalid reference to i
```

จากตัวอย่าง การอ้างถึง i ภายในนอกลูป for ไม่สามารถทำได้ เนื่องจากขอบเขตของตัวแปร i จะจำกัดอยู่เฉพาะภายในลูปเท่านั้น

ส่วน Dynamic scoping นั้น การ binding ชื่อหรือตัวแปรจะขึ้นอยู่กับลำดับการเรียกใช้ในโปรแกรม ไม่ใช่ตำแหน่งที่ประกาศ การค้นหาการประกาศจะค้นหาตามลำดับการเรียกใช้ ซึ่งจะตรวจสอบได้ขณะรันโปรแกรมเท่านั้น ดังตัวอย่าง

```
MAIN
  declaration of x
  SUB1
     declaration of x
     ...
     call SUB2
     ...
  SUB2
     ...
     reference to x
     ...
  ...
  call SUB1
  ...
```

จากตัวอย่าง โปรแกรมหลัก (MAIN) มีการประกาศตัวแปร x และโปรแกรมย่อย SUB1 ก็มีการประกาศตัวแปร x เช่นกัน โปรแกรมหลักมีการเรียกใช้ SUB1 ซึ่ง SUB1 ก็มีคำสั่งเรียกใช้โปรแกรมย่อย SUB2 ใน SUB2 มีการอ้างถึงตัวแปร x ถ้าภาษาโปรแกรมนี้ใช้ static scoping การอ้างถึงตัวแปร x ใน SUB2 จะอ้างถึง x ที่ประกาศไว้ในโปรแกรมหลัก แต่ถ้าใช้ dynamic scoping จะหมายถึง x ที่ประกาศไว้ใน SUB1 ตัวอย่างภาษาที่ใช้ dynamic scoping ได้แก่ Perl, LISP, SNOBOL4 เป็นต้น

10.3 ชนิดข้อมูล (Variables)

โดยทั่วไป ภาษาโปรแกรมมักจะประกอบด้วยชนิดข้อมูล 2 ประเภทหลัก คือ ชนิดข้อมูลพื้นฐาน (Primitive data types) และชนิดข้อมูลแบบที่ผู้ใช้กำหนด (User defined ordinal types)

ชนิดข้อมูลพื้นฐาน (Primitive data type)

โดยทั่วไป ภาษาโปรแกรมจะกำหนดให้มีกลุ่มของชนิดข้อมูลพื้นฐาน ซึ่งสามารถนำมารวมกันเพื่อสร้างเป็นชนิดข้อมูลแบบมีโครงสร้างได้ ชนิดข้อมูลพื้นฐานประกอบด้วย

- ตัวเลข (Numeric type) เช่น เลขจำนวนเต็ม (Integer) เลขจำนวนจริง (Floating-point) จำนวนจินตภาพ (Complex number)
- บูลีน (Boolean type) ซึ่งมีค่าเพียงสองค่าคือ จริง หรือ เท็จ เท่านั้น
- ตัวอักษร (Character type) ชนิดข้อมูลที่เก็บรหัสตัวเลขซึ่งใช้แทนตัวอักขระต่างๆ

ชนิดข้อมูลพื้นฐานชนิดเดียวกัน อาจมีชื่อเรียกต่างกันไปในแต่ละภาษา เช่น จำนวนเต็มในภาษา Pascal ใช้ integer ส่วนภาษา C ใช้ int เป็นต้น

ชนิดข้อมูลแบบที่ผู้ใช้กำหนด (User defined ordinal types)

ชนิดข้อมูลที่ผู้เขียนโปรแกรมสามารถกำหนดขึ้นใหม่ได้ โดยอาจจะเป็นการตั้งชื่อชนิดข้อมูลที่มีอยู่แล้วขึ้นใหม่ (rename) เช่นตัวอย่างในภาษา Pascal ต่อไปนี้

```
Type int = integer;
Var  a : int;
```

หรือเป็นชนิดข้อมูลที่กำหนดขึ้นเอง (enumeration) ดังเช่นตัวอย่างในภาษา C ต่อไปนี้

```
enum day {Mon, Tue, Wed, Thu, Fri, Sat, Sun};
```

หรืออาจเป็นชนิดข้อมูลที่มีโครงสร้าง (Structure data type) ซึ่งอาจจะประกอบด้วยชนิดข้อมูลพื้นฐานและชนิดข้อมูลที่กำหนดขึ้นเองมากระทำต่อกันด้วยวิธีการดังต่อไปนี้

- Cartesian products เป็นการนำชนิดข้อมูลที่เหมือนกันหรือต่างกันมาคูณกัน ผลลัพธ์ที่ได้จะเป็นชนิดข้อมูลชนิดใหม่ที่มีค่าของข้อมูลอยู่ในรูปคู่ลำดับ เช่น ชนิดข้อมูลประเภทเรคอร์ด (record) หรือโครงสร้าง (structure) ดังแสดงตัวอย่างในภาษา C ข้างล่างนี้

```
struct employee {
   int id;
   char name[20];
}
```

- Disjoint union เป็นการนำชนิดข้อมูลพื้นฐานที่ต่างกันมารวมกัน ผลลัพธ์ที่ได้จะเป็นชนิดข้อมูลที่มีค่าเป็นชนิดใดชนิดหนึ่งเท่านั้น เช่น ชนิดข้อมูลประเภท variant record หรือ union ดังตัวอย่างในภาษา C ต่อไปนี้

```
union myUnion {
   int i;
   float r;
};
union myUnion u;
```

จากตัวอย่างถ้าเราอ้างถึง u.i ก็จะเป็นตัวแปรที่มีชนิดข้อมูลเป็นจำนวนเต็ม ถ้าอ้างถึง u.r ก็จะเป็นตัวแปรที่มีชนิดข้อมูลเป็นจำนวนจริง

- Mapping เป็นการนำชนิดข้อมูลพื้นฐานชนิดเดียวกันมารวมกันเป็นกลุ่มโดยใช้ความสัมพันธ์แบบฟังก์ชัน ผลลัพธ์ที่ได้จะเป็นชนิดข้อมูลที่มีค่าของข้อมูลจากเซตหนึ่งไปยังค่าของข้อมูลอีกเซตหนึ่ง เช่น ชนิดข้อมูลแบบอาร์เรย์ (array) ซึ่งจัดเป็นชนิดข้อมูลที่ภาษาโปรแกรมส่วนใหญ่มีให้ใช้ ดังตัวอย่างในภาษา C ดังต่อไปนี้

```
int A[10];
float C[4][4];
char S[20];
```

- Powersets เป็นการนำชนิดข้อมูลชนิดเดียวกันมารวมเข้าด้วยกันคล้ายกับชนิดข้อมูลแบบอาร์เรย์ แต่การดำเนินการกับข้อมูลจะเหมือนกับการดำเนินการกับเซตในทางคณิตศาสตร์ เช่น union, intersection, different, subset ดังตัวอย่างในภาษา Pascal ต่อไปนี้

```
Type color = (red, green, blue);
     Hue = set of color;
```

ภาษาโปรแกรมโดยทั่วไปจะต้องมีการตรวจสอบชนิดข้อมูล (Type checking) ก่อนที่จะมีการดำเนินการ (operation) ใดๆ เพื่อป้องกันไม่ให้โปรแกรมทำงานที่เป็นไปไม่ได้กับชนิดข้อมูลหนึ่งๆ เช่น นำข้อมูลอักขระมาทำ operation คูณ กับข้อมูลบูลีน การตรวจสอบชนิดข้อมูลแบ่งเป็น 2 ประเภทตามช่วงเวลาที่ตรวจสอบ คือ

- Static type checking ตรวจสอบชนิดข้อมูลในช่วงเวลาการแปลโปรแกรม (compile time) โดยตรวจสอบว่าชนิดของตัวแปรที่กำหนดไว้ถูกต้องหรือไม่
- Dynamic type checking ตรวจสอบชนิดข้อมูลในช่วงเวลาการทำงานของโปรแกรม (run time) ตัวแปลภาษาจะไม่สามารถตรวจสอบได้ว่ากำหนดชนิดข้อมูลของตัวแปรถูกต้องหรือไม่ จนกว่าจะรันโปรแกรม

10.4 เอกสารอ้างอิงและเว็บไซต์ที่ควรรู้

Allen B. Tucker and Robert E. Noonan. Programming Languages – Principles and Paradigms.

Robert W. Sebesta. Concepts of Programming Languages.

สุจิตรา อดุลย์เกษม. เอกสารประกอบการสอนองค์ประกอบคอมพิวเตอร์และภาษา.