



# การเขียนโปรแกรมคอมพิวเตอร์ 1

## Computer Programming I

### โครงสร้างภาษาซี ตัวแปร และการแสดงผลอย่างง่าย

ภิญโญ แท้ประสาทสิทธิ์

Emails : pinyotae+111 at gmail dot com, pinyo at su.ac.th

Web : <http://www.cs.su.ac.th/~pinyotae/compro1/>

Facebook Group : [ComputerProgramming@CPSU](https://www.facebook.com/ComputerProgramming@CPSU)

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

สัปดาห์ที่สาม

## หัวข้อเนื้อหา



- โพลวชาร์ต ซูโดโค้ด และโค้ดภาษาคอมพิวเตอร์
- ภาษาซีกับการเขียนโปรแกรม
- การเริ่มต้นและการสิ้นสุดโปรแกรม
- ตัวแปร
  - ชนิดของข้อมูล
  - ความสัมพันธ์ระหว่างตัวแปรและชนิดของข้อมูล
  - การกำหนดค่าเริ่มต้นให้กับตัวแปร
  - ประเภทตัวแปร
- ค่าคงที่ในภาษาซี
- การแสดงผลอย่างง่าย

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

2

## โพลวชาร์ต ซูโดโค้ด และโค้ดภาษาคอมพิวเตอร์



- ของสามอย่างนี้เกี่ยวข้องกันเป็นอย่างยิ่ง
  - เราสามารถอ่านโพลวชาร์ตและซูโดโค้ดได้โดยไม่ต้องรู้ภาษาคอมพิวเตอร์
  - เราจึงเรียนของสองอย่างนี้ก่อน แล้วคิดแปลงมันให้เป็นภาษาคอมพิวเตอร์
- โพลวชาร์ตเขียนและเข้าใจได้ง่ายที่สุด
  - แต่ก็ต้องรู้เทคนิคในการแปลงเป็นโค้ดภาษาคอมพิวเตอร์ที่สอดคล้องกัน
  - คนจำนวนมากติดอยู่ตรงที่แปลงโพลวชาร์ตเป็นโค้ดไม่ได้
- ซูโดโค้ดจะมีความใกล้เคียงกับภาษาคอมพิวเตอร์มากกว่า
  - เป็นที่นิยมกว่าในการเรียนสำหรับชั้นปีต่อ ๆ ไป
  - เทคนิคหลายอย่างในซูโดโค้ดเหมือนกันกับในภาษาคอมพิวเตอร์เป๊ะ ๆ
- เราจะเขียนภาษาคอมพิวเตอร์ได้ดี ถ้าเราเขียนซูโดโค้ดได้ดี

16 กันยายน 2557

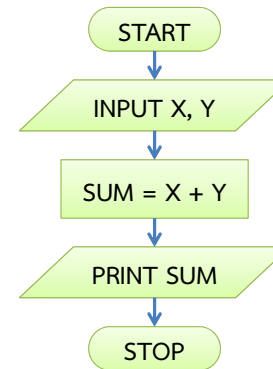
ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

3

## รู้จักกับการแปลงโพลวชาร์ตและซูโดโค้ดไปเป็นภาษาคอม



พิจารณาโพลวชาร์ตและซูโดโค้ดสำหรับการบวกเลขสองตัวต่อไปนี้



```

START
READ X
READ Y
COMPUTE SUM = X + Y
PRINT SUM
STOP
  
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

4

## แปลงเป็นโค้ดภาษาซี



คราวนี้ลองมาดูโค้ดภาษาซี

```
#include <stdio.h> // เตรียมตัวก่อนทำงาน
void main() {      // เริ่มการทำงาน (START)
    int x, y, sum;  // นิยามชื่อต่าง ๆ (ไม่ปรากฏในชุดโค๊ดหรือโฟลวชาร์ต)
    scanf("%d", &x); // อ่านค่า X (INPUT X)
    scanf("%d", &y); // อ่านค่า Y (INPUT Y)
    sum = x + y;    // หาค่าผลบวก แล้วเก็บไว้ใน sum (SUM = X + Y)
    printf("%d", sum); // แสดงผลบวกทางจอภาพ (PRINT SUM)
}                  // จบการทำงาน (STOP)
```

พบว่ามีของที่เพิ่มเข้ามาสองอย่างคือ ขั้นเตรียมตัวก่อนทำงานและนิยามต่าง ๆ

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

5

## ทบทวน : การคำนวณในคอมพิวเตอร์



- คนเราเรียนรู้และทำงานโดยอาศัยความรู้ความจำในอดีตเป็นตัวช่วย เช่น เราหาความยาวด้านสามเหลี่ยมจากมุมได้เพราะเรารู้เรื่องตรีโกณมิติ
- คอมพิวเตอร์เองก็เช่นกัน จะทำงานได้ก็ต้องการเก็บวิธีการคำนวณพื้นฐานต่าง ๆ ไว้ เช่น วิธีการคำนวณค่า sin, cos, และ tan
- เวลาคนเราคำนวณตัวเลข เราก็ต้องจำตัวเลขที่เกี่ยวข้องไว้ในหัวได้ เช่น “จงหาค่าของ  $5 + 3$ ” เราคำนวณได้ว่ามันมีค่าเท่ากับ 8
  - ถ้าเราลองทบทวนดูเราจะพบว่า ถ้าเราไม่สามารถจำเลข 5 และ 3 ไว้ในหัว เราได้เลยละก็ เราจะหาผลลัพธ์ออกมาไม่ได้เลย
- คอมพิวเตอร์ก็ต้องเก็บข้อมูลที่เกี่ยวข้องกับการคำนวณไว้ด้วย เช่น จากตัวอย่างเดิม เครื่องก็ต้องจำเลข 5 กับ 3 ไว้เพื่อใช้ในการหาผลบวก

8 พฤศจิกายน 2554

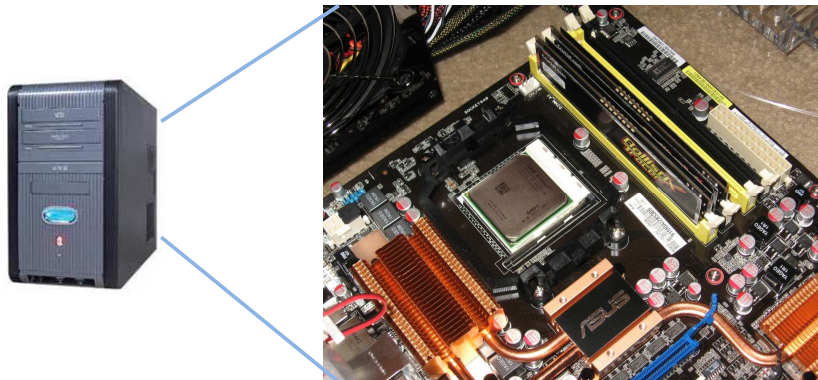
ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

6

## ทำไมต้องมีขั้นตอนแปลก ๆ เพิ่มเข้าด้วย



- ทบทวนของเก่า : ในขณะที่เครื่องทำการคำนวณ ซีพียู (CPU, หน่วยประมวลผลกลาง) จะมีการติดต่อกับหน่วยความจำ (Memory, RAM)
- ซีพียูกับหน่วยความจำเปรียบเหมือนสมองคนละส่วน : ส่วนความคิดและจดจำ



ภาพ overlockzone.com และ jedihawk.com

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

7

## การจดจำของคอมพิวเตอร์



มีอยู่สองส่วนหลัก

### 1. จดจำวิธีการทำงาน

ทำให้เราต้องบอกกับมันว่า #include <stdio.h> เพื่อบอกให้เครื่องรู้ว่าเราจะเรียกวิธีการทำงานแบบไหนออกมาใช้  
ในที่นี้คือการบอกให้เครื่องเตรียมพร้อมสำหรับการอ่านค่าจากผู้ใช้ และการแสดงผลบนจอภาพ นี่เป็นความจำขั้นพื้นฐานเปรียบได้กับการอ่านและการเขียนหนังสือของเรา

### 2. จดจำข้อมูลในการคำนวณ

เพราะคอมพิวเตอร์ต้องมีการจำข้อมูลเพื่อการคำนวณ ภาษาซีจึงต้องการให้เรานิยามค่าที่มันจะต้องจำเอาไว้ และนี่เป็นที่มาของบรรทัด int x, y, sum; เราบอกให้มันเตรียมจำค่าสามอย่างนี้ไว้

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

8

## ดูโค้ดภาษาซีอีกที



```
#include <stdio.h> // เตรียมตัวก่อนทำงาน
void main() {      // เริ่มการทำงาน (START)
    int x, y, sum;  // นิยามชื่อต่าง ๆ (ไม่ปรากฏในชุดโค๊ดหรือโฟลวชาร์ต)
    scanf("%d", &x); // อ่านค่า X (INPUT X)
    scanf("%d", &y); // อ่านค่า Y (INPUT Y)
    sum = x + y;    // หาค่าผลบวก แล้วเก็บไว้ที่ sum (SUM = X + Y)
    printf("%d", sum); // แสดงผลบวกทางจอภาพ (PRINT SUM)
}                  // จบการทำงาน (STOP)
```

- การเรียกความจำเกี่ยวกับวิธีการทำงาน ซึ่งก็คือ #include <stdio.h> มาก่อน ส่วนเริ่มทำงาน
- การเตรียมจำค่าต้องมาก่อนจังหวะที่ต้องจำค่าจริง ๆ

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

9

## แล้วทักษะในการบวกลบคูณหารของคอมล่ะ ?



**คำถาม** เห็นเห็นว่าเราต้องบอกเครื่องคอมให้เตรียมทักษะสำหรับการอ่านค่าและแสดงผล แล้วทำไมหนูไม่ต้องบอกให้มันเตรียมทักษะบวกลบคูณหารล่ะคะ ?

**คำตอบ** เพราะคอมพิวเตอร์เกิดมาเพื่องานการบวกลบคูณหารตั้งแต่แรก เป็นทักษะที่ติดตัวมาตั้งแต่คอมพิวเตอร์เกิด เหมือนคนเราเกิดมาปุ๊บก็ร้องไห้ได้เลย ค่ะ มันมีแม่ได้เลย เครื่องคอมเริ่มมาก็บวกลบคูณหารได้เลยอย่างนั้น ดังนั้นคนคิดค้นภาษาซีจึงเห็นว่าในเมื่อเครื่องคอมทำของแบบนี้ได้แต่แรก ก็ไม่ต้องคอยบอกเครื่องว่าให้เตรียมความสามารถพื้นฐานนี้ไว้ แต่ถ้าเป็นความยากระดับการหารากที่สอง (square root) หรือหาค่า sin, cos และ tan เราก็ต้องบอกมันเหมือนกันว่า #include <math.h>

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

10

## ความแตกต่างระหว่างภาษาซีกับโฟลวชาร์ตและชุดโค๊ด



- โฟลวชาร์ตและชุดโค๊ดเอาไว้บอกลำดับการคิดการทำงาน จะใช้กับอะไรก็ได้ ไม่จำเป็นต้องเป็นเครื่องคอม
  - การจำและการเตรียมทักษะเป็นสิ่งที่ไม่ต้องเขียนไว้ในโฟลวชาร์ตและชุดโค๊ด
- แต่ภาษาซีเอาไว้ใช้กับเครื่องคอม จึงจำเป็นที่จะต้องมีส่วนเพิ่มเติม
  - เป็นไปเพื่อความสอดคล้องกับการทำงานของเครื่อง
  - เพราะเราต้องการสื่อสารกับเครื่อง ก็ต้องเขียนแบบที่เครื่องคอมจะเข้าใจ
- ความแตกต่างเหล่านี้เป็นสิ่งที่เราต้องจดจำเอาไว้ พลาดไม่ได้เด็ดขาด

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

11

## การจดจำค่าในภาษาซี



- เราได้ฝึกฝนกระบวนการวิเคราะห์ปัญหา ตลอดจนการเขียนโฟลวชาร์ตและชุดโค๊ดมาพอสมควรแล้ว
- เราต้องการแปลงโฟลวชาร์ตและชุดโค๊ดไปเป็นภาษาซี
- แต่เราติดตรงที่ว่าสิ่งที่จำเป็นในภาษาซีบางอย่าง ไม่มีอยู่ในทั้งโฟลวชาร์ตและชุดโค๊ด
  - นั่นคือส่วนที่เป็นการเตรียมทักษะและความสามารถบางอย่าง อันนี้ทำได้ง่าย เพราะความสามารถที่ต้องบอกเครื่องให้เตรียมไว้มีอยู่ไม่กี่แบบ และมีลักษณะตายตัว
  - แต่การจดจำค่าที่ดูเหมือนง่าย กลับมีรูปแบบและวิธีการใช้งานที่หลากหลาย
  - ทำให้เราต้องศึกษาวิธีการจดจำค่าในภาษาซี เพื่อการแปลงแนวคิดที่มีในโฟลวชาร์ตและชุดโค๊ดให้เป็นโปรแกรมภาษาซีที่ถูกต้องได้

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

12

## ตัวแปร (Variable)



**ตัวแปร** คือ สิ่งที่ใช้เก็บข้อมูลในโปรแกรมคอมพิวเตอร์ทั้งตัวเลขและตัวอักษร

- ต่างกับตัวเลขทั่วไปตรงที่ว่าตัวแปรจะมีชื่อกำกับ เช่น  $x$ ,  $y$ ,  $m$ , และ  $n$
- เราสามารถกำหนดค่าให้กับตัวแปรต่าง ๆ ได้ ผ่านชื่อของมัน
  - ในการกำหนดค่าชื่อตัวแปรต้องอยู่ด้านซ้ายของค่าที่จะใส่
  - เช่น  $x = 3$ ;  $y = -12$ ;
- เราสามารถคัดลอก (copy) ค่าจากตัวแปรหนึ่ง ไปอีกตัวแปรหนึ่งได้
  - ตัวแปรที่จะถูกกำหนดค่าต้องอยู่ด้านซ้าย
  - เช่น  $x = y$ ; แปลว่า ‘คัดลอกค่าจาก  $y$  ไปเก็บไว้ที่  $x$ ’
  - ผลลัพธ์จะทำให้ทั้ง  $x$  และ  $y$  มีค่าเท่ากับ  $-12$  (ไม่ใช่  $3$ )
- ตัวแปรเมื่ออยู่ด้านซ้ายและขวาจะมีบทบาทหน้าที่ต่างกัน

## ชนิดข้อมูล (Data Type)



**ชนิดข้อมูล** ทำหน้าที่ระบุวิธีจัดจำข้อมูลในหน่วยความจำ

- ตัวแปรทุกตัวทุกชนิดถูกเก็บไว้ในหน่วยความจำ แต่วิธีเก็บก็แตกต่างกัน
- ความแตกต่างนี้ส่งผลต่อขอบเขตค่าที่จัดจำได้รวมทั้งวิธีตีความหมายตัวแปร
- การระบุชนิดข้อมูลให้สอดคล้องกับค่าที่ต้องการจัดจำเป็นเรื่องที่สำคัญมาก
  - จุดนี้หลายคนทำผิด เพราะกระบวนการคิดในหัวเราไม่ต้องคำนึงถึงจุดนี้โดยรวม

**ชนิดข้อมูลของตัวแปรมีอยู่สองกลุ่ม**

1. **กลุ่มพื้นฐาน (Basic Data Type)** มีอยู่สี่แบบคือ (1) แบบจำนวนเต็ม, (2) แบบเลขทศนิยม, (3) แบบตัวอักษร, และ (4) แบบ void
2. **กลุ่มขั้นสูง (Advanced Data Type)** เช่น อาเรย์, ตัวชี้ และ สตริค เราจะเรียนชนิดข้อมูลกลุ่มนี้ในครั้งหลัง

## ชนิดข้อมูลแบบจำนวนเต็ม



- เป็นชนิดข้อมูลที่พบบ่อยที่สุด แบบที่เราพบบ่อยในปัจจุบันคือ
  - แบบ int เป็นเลขจำนวนเต็มมาตรฐาน เพราะใช้หน่วยความจำไม่มาก (4 bytes) และเก็บค่าได้ตั้งแต่  $-2,147,483,648$  ถึง  $2,147,483,647$
  - แบบ short int เป็นเลขจำนวนเต็มแบบประหยัดหน่วยความจำ คือใช้พื้นที่หน่วยความจำแค่ 2 bytes เท่านั้น แต่ค่าที่เก็บได้ก็น้อยไปด้วย คือเก็บได้แค่จาก  $-32,768$  ถึง  $32,767$
  - ในสมัยที่คอมพิวเตอร์มีหน่วยความจำน้อย int ใช้พื้นที่แค่ 2 bytes และทำตัวเหมือน short int ทุกอย่าง (พบได้ใน Turbo C)
  - ในปัจจุบันเครื่องคอมพิวเตอร์ดีขึ้น มาตรฐานภาษาซีจึงถูกยกระดับขึ้นมาเป็นอย่างที่ระบุไว้ในนี้

## ขนาดของตัวแปร



- เราได้กล่าวถึงปริมาณหน่วยความจำที่ใช้เก็บตัวแปร int และ short int มาแล้ว ปริมาณหน่วยความจำที่ใช้ันี้มักถูกเรียกว่า *ขนาดของตัวแปร*
- นิยาม ขนาดของตัวแปร (Variable Size)** คือ ปริมาณหน่วยความจำที่ใช้ในการเก็บตัวแปรนั้น ๆ
- เราได้กล่าวถึงขนาดของ int ที่เปลี่ยนไปตามกาลเวลา เรื่องนี้สร้างความสับสนเล็กน้อยให้กับมือใหม่ ในหนังสือเรียนบทที่ 5 หน้า 58 จึงมีการระบุขนาดของ int ไว้ทั้งที่เป็น 4 bytes (แบบปัจจุบัน) และ 2 bytes (แบบโบราณ)
- เราจะถือตามแบบปัจจุบันซึ่งสอดคล้องกับ Code::Blocks
  - นั่นคือ int มีขนาดเท่ากับ 4 bytes ซึ่งเท่ากับ 32 bits (1 byte มี 8 bits)

## ตัวอย่างการใช้งาน



```
#include <stdio.h>
void main() { // เริ่มการทำงาน (START)
    int x, y, sum; // ประกาศตัวแปรด้วยการระบุชนิดข้อมูลแล้วตามด้วยชื่อ
    x = 5; // กำหนดค่าของ x ให้เป็นไปตามเลขทางขวา
    y = 7; // กำหนดค่าของ y ให้เป็นไปตามเลขทางขวา
    sum = x + y; // หาค่าผลบวก แล้วเก็บไว้ที่ sum (ได้ผลเป็น 12)
    printf("%d", sum); // แสดงผลบวกทางจอภาพ (ได้เลข 12 ที่จอภาพ)
} // จบการทำงาน (STOP)
```

ตัวแปร x, y ตอนอยู่ด้านซ้ายจะถูกเขียนค่า

แต่พอ x, y มาอยู่ทางขวาจะเป็นการอ่านค่าที่มันเก็บไว้

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

17

## ประกาศตัวแปรแยกกันก็ได้



โค้ดนี้ให้ผลลัพธ์เหมือนกับโค้ดในหน้าที่แล้วทุกประการ

```
#include <stdio.h>
void main() {
    int x;
    int y;
    int sum;
    x = 5;
    y = 7;
    sum = x + y;
    printf("%d", sum);
}
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

18

## จะประกาศตัวแปรพร้อมกำหนดค่าเลยก็ได้



โค้ดนี้ให้ผลลัพธ์เหมือนกับโค้ดที่แสดงมาก่อนหน้านี้ทุกประการ

```
#include <stdio.h>
void main() {
    int x = 5;
    int y = 7;
    int sum = x + y;
    printf("%d", sum);
}
```

การกำหนดค่าเริ่มต้นจะเป็นตัวเลขหรือตัวแปรก็ได้

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

19

## เครื่องหมายและความเป็นบวกลบของเลขจำนวนเต็ม



- ในบางครั้งเราใช้เลขจำนวนเต็มเพื่อระบุจำนวนสิ่งของหรือบุคคล
- เนื่องจากจำนวนเหล่านี้มีค่าตั้งแต่ 0 ขึ้นไป ไม่มีค่าติดลบ จึงมีความคิดที่จะตัดค่าติดลบออกไปเพื่อให้สอดคล้องกับการใช้งาน
- ชนิดข้อมูลจำนวนเต็มจึงมีค่าพิเศษเพิ่มขึ้นมาคือ unsigned เพื่อระบุว่าตัวแปรจำนวนเต็มนี้จะมีเฉพาะค่า 0 กับค่าบวกเท่านั้น
  - เช่น unsigned int เก็บค่าตั้งแต่ 0 ถึง 4,294,967,295 ในขณะที่ signed int เก็บค่าตั้งแต่ -2,147,483,648 ถึง 2,147,483,647
  - ข้อมูลแบบ unsigned ทำให้ตัวแปรเก็บค่าบวกได้มากขึ้น
  - หมายเหตุ เรานิยมเขียนประเภทข้อมูล signed int แบบสั้นว่า int กล่าวคือเราเขียนว่า int x; แทนที่จะเขียนว่า signed int x;

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

20

## แล้วขนาดของ unsigned int ละ



- แท้จริงแล้ว unsigned int กับ int มีขนาดเท่ากัน คือมีขนาด 32 bits ด้วยกันทั้งคู่ ลองสังเกตดูจากตรงนี้ :
  - unsigned int เก็บค่าตั้งแต่ 0 ถึง 4,294,967,295 และ int เก็บค่าตั้งแต่ -2,147,483,648 ถึง 2,147,483,647
  - ลองคำนวณความแตกต่างของค่าที่ int จัดจำได้ 2,147,483,647 - (-2,147,483,648) = 4,294,967,295
  - แสดงว่าความกว้างของช่วงค่าที่จัดจำได้ทั้งแบบที่มีเครื่องหมาย (signed) และแบบไม่มีเครื่องหมาย (unsigned) มีค่าเท่ากัน
  - ที่เป็นแบบนี้เพราะพื้นที่เก็บข้อมูลจะเป็นตัวระบุจำนวนตัวเลขที่แสดงได้

## ตัวอย่างเปรียบเทียบ signed และ unsigned



```
void main() {
    int s_int1 = 5;           // ใช้ได้ ไม่มีปัญหา
    unsigned int u_int1 = 5; // ใช้ได้ ไม่มีปัญหา
    int s_int2 = -5;         // ใช้ได้ เพราะ int เก็บค่าติดลบได้
    unsigned int u_int2 = -5; // ใช้ไม่ได้ เพราะ unsigned ไม่เก็บค่าติดลบ
                                // ทำแบบนี้ค่าจะไม่ออกมาอย่างที่ควรเป็น
    int s_int3 = 4000000000; // ใช้ไม่ได้ เพราะ int เก็บค่าบวกได้มากที่สุด
                                // สุดประมาณสองพันล้านเท่านั้น
    unsigned int u_int3 = 4000000000; // ใช้ได้เพราะเป็นค่าบวก
                                        // และไม่เกิน 4,294,967,295
}
```

## สรุปเกี่ยวกับข้อมูลประเภทจำนวนเต็ม



- สองแบบที่ใช้บ่อยคือ
  - int ซึ่งนิยมมากที่สุด และใช้พื้นที่ 4 bytes หรือ 32 bits (นับขนาดตามรูปแบบปัจจุบัน ซึ่งเป็นแบบเดียวกับที่เราใช้ในการทำเล็บ)
  - short int ถูกลำเอียงใช้เมื่อต้องการประหยัดหน่วยความจำ
- เราแบ่งการเก็บจำนวนเต็มเป็นแบบที่มีและไม่มีเครื่องหมาย
  - ในกรณีที่ต้องการเก็บแต่เลขศูนย์และค่าบวก เราใส่คำนำหน้าว่า unsigned
  - ถ้าหากจะให้เก็บค่าลบได้ด้วย เราไม่จำเป็นต้องใส่คำนำหน้า หรือถ้าจะใส่ก็ใส่คำว่า signed (แต่ไม่นิยมใส่)

## เรื่องน่ารู้เกี่ยวกับตัวแปรแบบจำนวนเต็ม



- แม้เราต้องการจะเก็บแต่ค่าบวกเพียงอย่างเดียว เราก็มักจะเลือกใช้ int
  - ขอแค่ค่าบวกที่มากที่สุดไม่เกินสองพันล้านการใช้ int จะไม่ทำให้ผลลัพธ์ผิด
  - ข้อมูลส่วนใหญ่มีค่าไม่ถึงสองพันล้าน
- ยังมีตัวแปรแบบจำนวนเต็มอื่น ๆ เช่น long long int ซึ่งเป็นมาตรฐานใหม่
  - คอมไพเลอร์บางตัวยังไม่รองรับมาตรฐานใหม่นี้
  - ใช้พื้นที่เก็บข้อมูลมากถึง 8 bytes หรือ 64 bits
  - เก็บข้อมูลได้กว้างมาก คือ -9,223,372,036,854,775,808 ถึง 9,223,372,036,854,775,807 (เก้าล้านล้านล้าน)
  - พบได้ในการคำนวณทางวิทยาศาสตร์และการเงิน (ไม่มีการใช้ในวิชา)

## สรุปเกี่ยวกับการประกาศตัวแปร



- เป็นการบอกให้คอมพิวเตอร์รู้ว่าจะต้องจำข้อมูลอะไรบ้างในการคำนวณ
- การประกาศจะเริ่มด้วยชนิดข้อมูลที่ต้องการ และตามด้วยชื่อของตัวแปร
  - เช่น short nBooks; เป็นการระบุให้เครื่องเก็บตัวแปรจำนวนเต็ม 2 bytes แบบมีเครื่องหมายบวกลบ โดยใช้ชื่อตัวแปรว่า nBooks (ชื่อตัวแปรจะเป็นตัวอักษรโดดหรือประกอบด้วยตัวอักษรหลายตัวก็ได้)
  - เช่น int a; หรือ unsigned int ant; เป็นการระบุให้เครื่องเก็บตัวแปรจำนวนเต็ม 4 bytes โดย a จะเก็บค่าลบได้ ส่วน ant เก็บค่าลบไม่ได้
- ถ้าตัวแปรหลายตัวมีชนิดข้อมูลเดียวกัน เราสามารถประกาศตัวแปรเหล่านี้พร้อมกันก็ได้ เช่น int a, b, c; เป็นการประกาศตัวแปร int สามตัวพร้อมกัน
- สามารถประกาศพร้อมระบุค่าได้ เช่น int x = 7; และ int sum = x + 25;

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

25

## กฎเหล็กเกี่ยวกับการตั้งชื่อตัวแปร



1. ชื่อตัวแปรมีของปนกันได้อยู่สามอย่างคือ (1) ตัวอักษรภาษาอังกฤษ, (2) ตัวเลข และ (3) เครื่องหมายขีดเส้นใต้ (เครื่องหมาย \_)
  - เช่น int s\_int1; เป็นการกำหนดให้ชื่อตัวแปรเป็น s\_int1
  - เช่น int ab\_12\_k; เป็นการกำหนดให้ชื่อตัวแปรเป็น ab\_12\_k
2. **ห้าม**ชื่อตัวแปรขึ้นต้นด้วยตัวเลข เช่น int 1s\_int; **แบบนี้ใช้ไม่ได้** เพราะเอาตัวเลขขึ้นต้นชื่อตัวแปร
3. ชื่อขึ้นต้นด้วยเครื่องหมายขีดเส้นใต้ได้ เช่น int \_foo; แบบนี้ใช้ได้
4. ห้ามตัวแปรสองตัวในเขตพื้นที่เดียวกัน ชื่อซ้ำกัน ไม่ว่าจะตัวแปรสองตัวนั้นจะมีประเภทข้อมูลเหมือนหรือแตกต่างกัน เช่น int x; float x; แบบนี้ถือว่าผิดกฎเพราะชื่อตัวแปรซ้ำกัน

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

26

## ชื่อตัวแปรกับอักษรตัวเล็กตัวใหญ่



- ในภาษาซี ตัวอักษรเล็กใหญ่มีความสำคัญ เพราะถือว่าเป็นคนละตัวกัน
  - ชนิดข้อมูลต้องสะกดตามที่เห็นในหนังสือ เช่น เราต้องใช้คำว่า int ซึ่งเป็นตัวเล็กทั้งหมด จะใช้คำว่า INT หรือ Int แบบนี้ไม่ได้
  - ชื่อตัวแปรก็เช่นกัน แม้จะต่างกันแค่ตัวเล็กตัวใหญ่ก็ถือว่าต่างกัน เช่น int x; กับ int X; ถือว่าเป็นคนละตัวคนละชื่อกัน นี่เป็นกฎเกี่ยวกับชื่อตัวแปรที่สำคัญมากอีกข้อหนึ่ง
- ผู้เริ่มเรียนมักสับสนกับเรื่องตัวเล็กตัวใหญ่ของชื่อตัวแปร ทำให้โปรแกรมคอมไพล์ไม่ผ่าน เช่น ตอนประกาศ เขียนว่า int Number; แต่พอเอาไปใช้ ไปเขียนว่า number = 5; แบบนี้ก็ไม่ได้ คอมไพล์ไม่ผ่าน

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

27

## จำเป็นหรือไม่ที่จะต้องใช้ตัวแปรในการคำนวณทุกครั้ง ?



- **ไม่จำเป็น** ในกรณีที่เรารู้ตัวเลขที่แน่นอนตั้งแต่ตอนเขียนโปรแกรม เราสามารถระบุตัวเลขลงไปในการคำนวณได้เลย
  - เช่น ในตัวอย่างเราเขียนว่า int sum = x + y; ถ้าเรารู้ค่า x และ y ตั้งแต่ตอนเขียนโปรแกรม เราเขียนว่า int sum = 5 + 7; ไปเลยก็ได้
  - ถึงไม่จำเป็นแต่บางทีก็ควรใช้ เพราะค่าคงที่บางอย่างใช้บ่อยและเขียนยาก เช่น ค่า  $\pi \approx 3.1415926535$  ถ้าหากเราจะเขียนเลขนี้ทุกครั้งที่โค้ดเราจะดูยุ่งเหยิงและเสี่ยงที่จะพิมพ์ผิดไปเป็นค่าอื่นได้
- **จำเป็น** ในกรณีที่มีการรับข้อมูลจากผู้ใช้ เราต้องส่งตัวแปรไปรับค่าจากผู้ใช้มา ความจำเป็นนี้เกิดขึ้นเพราะเราไม่ทราบค่าของตัวเลขจนกว่าผู้ใช้จะระบุมาในภายหลัง

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

28

## ชนิดข้อมูลพื้นฐาน 4 แบบ



จัดตามลักษณะข้อมูลที่เก็บ ชนิดข้อมูลพื้นฐานมีอยู่สี่แบบดังนี้

1. แบบจำนวนเต็ม
2. แบบเลขทศนิยม
3. แบบตัวอักษร
4. แบบ void

## ชนิดข้อมูลเลขทศนิยม



- เลขทศนิยมที่เรารู้จักมักมีหน้าตาในการทำงานที่ 12.3456 หรือ -7.52
- คำถามที่น่าสนใจก็คือว่า คอมพิวเตอร์จะเก็บค่าเหล่านี้ไว้อย่างไร
  - โดยเฉพาะพวกทศนิยมไม่รู้จบเช่น 3.777777... แบบนี้จะทำอย่างไร
- มีมาตรฐานการเก็บเลขทศนิยมในคอมพิวเตอร์
  - มีสองแบบที่ใช้อยู่คือ แบบความเที่ยงเดี่ยว (*single precision*) และแบบความเที่ยงทวิคูณ (*double precision*)
  - ทั้งแบบ single และ double precision มีแนวคิดการเก็บเหมือนกัน คือ แบ่งพื้นที่ในการจำค่าเป็นสามส่วน : (1) เครื่องหมายบวกลบ (sign) , (2) เลขนัยสำคัญ (mantissa) และ (3) เลขชี้กำลัง (exponent)
  - แบบ double precision จะเก็บข้อมูลทั้งเลขนัยสำคัญและเลขชี้กำลังได้ดีกว่าแบบ single precision แต่ก็ใช้พื้นที่ในหน่วยความจำมากกว่าเท่าตัว

## ชนิดข้อมูลแบบเลขทศนิยมและขอบเขตของค่า



- ชนิดข้อมูลแบบเลขทศนิยมแบบ single precision และ double precision ในภาษาซี มีชื่อว่า float และ double
  - เช่น float  $x = 3.5$ ; และ double  $y = -123.456$ ;
- ขอบเขตของค่าที่ชนิดข้อมูลแบบ float เก็บได้มีค่าดังต่อไปนี้
  - ในกรณีค่าบวก ค่าที่น้อยที่สุดคือ  $3.4 \times 10^{-38}$  และค่ามากที่สุดคือ  $3.4 \times 10^{38}$
  - ค่าลบก็เป็นไปในทำนองเดียวกันคือเก็บได้ตั้งแต่  $-3.4 \times 10^{-38}$  ถึง  $-3.4 \times 10^{38}$
- ขอบเขตของค่าที่ชนิดข้อมูลแบบ double เก็บได้มีค่าประมาณดังนี้
  - ในกรณีค่าบวก ค่าที่น้อยที่สุดคือ  $1.7 \times 10^{-308}$  และค่ามากที่สุดคือ  $1.7 \times 10^{308}$
  - ส่วนค่าลบจะอยู่ในช่วง  $-1.7 \times 10^{-308}$  ถึง  $-1.7 \times 10^{308}$

## ข้อจำกัดของตัวแปรแบบ float และ double



- ความแตกต่างของ float และ double ไม่ได้อยู่ที่เฉพาะค่าสูงสุดหรือต่ำสุด แต่อยู่ที่ความเที่ยงของข้อมูลด้วย
- ชนิดข้อมูลแบบ float มีความเที่ยงของข้อมูลค่อนข้างน้อย โดยแสดงเลขฐานสิบโดยถูกต้องได้เพียงประมาณ 7 หลักเท่านั้น ในขณะที่ int ทำได้ถึง 9 หลัก
- เรื่องที่น่าสนใจก็คือว่า float ใช้พื้นที่ทั้งหมด 4 ไบต์เท่ากับ int
- แต่เลข  $3.4 \times 10^{38}$  มีค่ามากกว่าสองพันล้าน ( $2 \times 10^9$ ) ที่ตัวแปรแบบ int เก็บได้อย่างเห็นได้ชัด เพราะ float ยอมลดความเที่ยงของข้อมูล เพื่อให้บันทึกตัวเลขค่ามาก ๆ ได้
- ข้อมูลแบบ double มีความเที่ยงสูงมาก สูงยิ่งกว่า int เพราะความเที่ยงของข้อมูลมีถึง 15 หลัก แต่ double ต้องการพื้นที่มากถึง 8 ไบต์



## การใช้งาน float และ double



- เป็นไปในทำนองเดียวกับข้อมูลชนิดจำนวนเต็ม คือ
  - การประกาศต้องขึ้นด้วยชนิดข้อมูล ตามด้วยชื่อตัวแปร
  - สามารถกำหนดค่าเริ่มต้นได้ตอนกำหนดชื่อตัวแปร
  - ถ้าชนิดข้อมูลเป็นแบบเดียวกัน สามารถประกาศชื่อตัวแปรหลายตัวพร้อมกันได้
- ความแตกต่างของการใช้งานจะเกิดขึ้นตอนดำเนินการทางเลขคณิตและตอนแสดงผล ซึ่งเราจะเรียนเรื่องนี้โดยละเอียดในภายหลัง

## ชนิดข้อมูลพื้นฐาน 4 แบบ



จัดตามลักษณะข้อมูลที่เก็บ ชนิดข้อมูลพื้นฐานมีอยู่สี่แบบนี้

1. แบบจำนวนเต็ม
2. แบบเลขทศนิยม
3. แบบตัวอักษร
4. แบบ void

## ชนิดข้อมูลตัวอักษร



- เป็นข้อมูลอีกชนิดที่พบบ่อย เพราะมักเกี่ยวข้องกับการแสดงผล
- ข้อมูลชนิดนี้แทนอักขระหนึ่งตัว ไม่ได้แทนข้อความทั้งข้อความ
- การประกาศตัวแปรชนิดนี้ให้ใช้คำว่า char เช่น  
char letter = 'x'; เป็นการประกาศตัวแปรชื่อ letter ให้เป็นอักขระตัวเอ็กซ์เล็ก
  - ต้องใช้เครื่องหมายอัฒภาคเดี่ยว (single quotes) ครอบตัวอักขระที่เราต้องการไว้
  - สาเหตุที่ต้องใช้ single quotes ก็คือมันทำให้คอมไพเลอร์แยกได้ว่าตัวเอ็กซ์ที่เห็นนี้เป็นตัวอักขระ ไม่ใช่ชื่อตัวแปร

## เรื่องทางเทคนิคเกี่ยวกับ char



- char ใช้พื้นที่ในหน่วยความจำ 1 ไบต์
- ตัวจริงของ char เก็บเลขจำนวนเต็มไว้ แต่คอมไพเลอร์ตีความหมายของจำนวนเต็มออกมาเป็นตัวอักษรตามมาตรฐาน ASCII (อ่านว่า แอสกี) ซึ่งย่อมาจาก American Standard Code for Information Interchange
- ที่ต้องมีรหัสแบบนี้ขึ้นมาก็เพราะว่าเครื่องคอมพิวเตอร์เก็บได้แต่ตัวเลข ไม่ได้เก็บตัวอักษร จึงมีการสร้างข้อตกลงร่วมกันกำหนดให้ตัวเลขแต่ละตัวแทนตัวอักษรต่าง ๆ กันไป เช่น เลข 97 แทน a และเลข 98 แทน b
- เพราะข้างใน char เป็นตัวเลข ดังนั้นเราจึงสามารถทำการบวกลบค่าของตัวแปร char เพื่อให้เปลี่ยนเป็นตัวอักษรอื่นได้ ดังแสดงในตัวอย่างหน้าถัดไป

## ตัวอย่าง 5.1 การดำเนินการคณิตศาสตร์กับตัวอักษร



```
#include <stdio.h>
void main() {
    char A = 'a';
    printf("Variable A is %c\n", A);
    A = A + 10;
    printf("After plus variable A with 10, variable A is %c\n", A);
}
```

ผลลัพธ์ของโปรแกรมจะเป็นข้อความบนหน้าจอดังนี้

```
Variable A is a
After plus variable A with 10, variable A is k
```

## อธิบาย : ทำไม a ถึงเปลี่ยนเป็น k



การเปลี่ยนแปลงเกิดขึ้นเพราะว่าอักษร 'a' นั้นมีค่าตัวเลขที่เครื่องคอมพิวเตอร์เก็บไว้เป็น 97 ซึ่งเป็นไปตามรหัสแอสกี (ดูหนังสือหน้า 336 ประกอบ)

- นั่นคือ ตัวแปร A แท้จริงข้างในเก็บเลข 97 เอาไว้
- เมื่อโปรแกรมดำเนินการ  $A = A + 10$ ; ค่าตัวแปร A ใหม่ที่ได้จะเป็น 107
- เลข 107 นี้ตรงกับค่ารหัสของ k (ดูหนังสือหน้า 336 ประกอบ)
- ดังนั้นเมื่อเครื่องคอมพิวเตอร์ทำการแปลงค่า 107 ในตัวแปร A มาเป็นตัวอักษร เครื่องจะแสดงตัวอักษร k ออกมา

## อักขระพิเศษ



- ข้อมูลแบบ char ไม่ได้เก็บเพียงแค่ตัวอักษร a-z หรือ A-Z ไว้
  - มันยังเก็บเครื่องหมายวรรคตอน เช่น ? ! : ; , ได้ด้วย
  - ตัวเลขและเครื่องหมายคณิตศาสตร์ที่มีบนแป้นพิมพ์ก็เก็บได้
  - รวมถึงเครื่องหมายพิเศษอื่น ๆ เช่น @ \$ % ^ & \* เป็นต้น
- แต่ทั้งนี้ก็ยังมียกขระพิเศษอีกหลายตัว ที่ไม่ใช่ตัวอักษรเสียทีเดียวที่ char เก็บไว้ เช่น การขึ้นบรรทัดใหม่ (new line), การเลื่อนไปตำแหน่งกั้นหน้าถัดไป (tab), และ เสียงเตือนจากเครื่อง (bell)
  - ของเหล่านี้เรียกว่าตัวอักขระพิเศษ ซึ่งเราสามารถอ้างถึงอักขระพวกนี้ได้ผ่านการใช้เครื่องหมาย \ (backslash) แล้วตามด้วยรหัสที่แทนอักขระพิเศษ

## อักขระพิเศษยอดนิยม



- การขึ้นบรรทัดใหม่ แทนด้วย \n
- การเลื่อนไปตำแหน่งกั้นหน้าถัดไป (tab) แทนด้วย \t
- จุดสิ้นสุดข้อความ (null) แทนด้วย \0
- เสียงเตือน (bell) แทนด้วย \a

เช่น `printf("a\tb\nca");` จะพิมพ์ข้อความ

```
a      b
c
```

บนจอแล้วตามด้วยเสียงเตือนหนึ่งครั้ง

ทั้งนี้เป็นผลมาจาก a ตามด้วย Tab (\t) จากนั้นตามด้วย b และการขึ้นบรรทัดใหม่ (\n) จากนั้นตามด้วย c และปิดท้ายด้วยเสียงเตือน (\a)



## อักขระหลีก (Escaped Character)



**คำถาม** ก็ในเมื่อเราใช้ \ ไปเพื่อจัดการอักขระพิเศษ เช่น \n แทนการขึ้นบรรทัดใหม่ ถ้าหากเราอยากพิมพ์ \n จริง ๆ บนหน้าจอล่ะ คืออยากพิมพ์ตัว \ แล้วตามด้วยตัว n บนหน้าจอ ไม่ต้องการขึ้นบรรทัดใหม่ จะทำอย่างไร ?

**คำตอบ** เราต้องใช้อักขระหลีกสำหรับตัวอักษรที่มีหน้าที่หลายอย่างในภาษาซี

- เครื่องหมาย backslash แทนด้วย \\ (backslash ติดกันสองตัว)
- อัญประกาศคู่ (“) แทนด้วย \" (แก้ปัญหาความซ้ำซ้อนของการใช้อัญประกาศคู่เพื่อระบุข้อมูลแบบสตริง)

ดังนั้นจากคำถามด้านบน ถ้าเราอย่างพิมพ์ข้อความ \n บนจอภาพเราต้องใช้ `printf(“\\n”);`

## char กับข้อความที่มีหลายตัวอักษร



**คำถาม** ในเมื่อ char เป็นแค่ตัวอักษรโดด แล้วเราจะจัดการกับข้อความที่มีหลายตัวอักษรได้อย่างไร ?

**คำตอบ** เราต้องนำตัวอักษรโดดมาต่อกันเพื่อให้ได้ข้อความที่ต้องการ ชนิดข้อมูลที่เก็บข้อความนั้นเรียกว่า สตริง (string) ซึ่งเป็นโครงสร้างข้อมูลชั้นสูงเพราะประกอบขึ้นมาจากแถวลำดับ (อาเรย์; array)

\*\* สตริง มีชื่ออย่างเป็นทางการในภาษาไทยว่า ‘สายอักขระ’ แต่หากใช้คำว่าสายอักขระในการสื่อสาร คนจำนวนมากจะรู้สึกว่าเป็นธรรมชาติ ดังนั้นเราจะใช้คำว่าสตริงเป็นหลัก

## พื้นฐานน่ารู้เกี่ยวกับสตริง



- เวลาเราใช้คำสั่ง `printf` เพื่อแสดงข้อความบนหน้าจอ แท้จริงแล้วเราส่งสตริงไปให้เครื่องแสดงผลออกมา เช่น `printf(“Welcome to Silpakorn”);`
  - เราใช้ double quotes ครอบข้อความที่ต้องการพิมพ์ไว้
  - การใช้ double quotes คือการเปลี่ยนชุดตัวอักษรให้เป็นสตริงในภาษาซี
  - อย่าสับสนกับการใช้ single quotes กับตัวอักษรโดดเป็นอันขาด

**คำถาม** แล้วถ้าจะแสดงตัวอักษรโดดออกมาทางจอภาพล่ะ เราควรใช้ single หรือ double quotes ?

**คำตอบ** ในเมื่อการแสดงผลรับเฉพาะข้อมูลแบบสตริง แม้จะเป็นตัวอักษรโดดก็ต้องใช้ double quotes เช่น `printf(“a”);` เป็นต้น

## ชนิดข้อมูลแบบ void



`void` เป็นประเภทของตัวแปรพื้นฐาน แต่การใช้งานก็ถือว่าหลากหลายมาก คือ มันถูกใช้ระบุว่า ‘ไม่มีชนิดข้อมูล’ หรือ ‘ยังไม่ระบุชนิดข้อมูล’ เรามักพบการใช้ `void` กับฟังก์ชัน เช่น ในส่วนบอจุดเริ่มต้นของโปรแกรม เราเขียนว่า

```
void main( ) {
```

```
...
```

```
}
```

เพราะฟังก์ชันสามารถทำตัวคล้ายตัวแปรได้ จึงต้องมีการระบุชนิดข้อมูลกำกับ แต่ในกรณีที่เราไม่ต้องการให้มันทำหน้าที่คล้ายตัวแปร เราจะใช้คำว่า `void` นำหน้า เพื่อบอกว่า ‘ไม่มี’ ชนิดข้อมูลที่อยากให้ `main` เป็น

## สรุปกฎพื้นฐานของการใช้ตัวแปร (1)



- ต้องมีการประกาศตัวแปรก่อนเสมอ โดยการประกาศอยู่ในรูปมาตรฐาน : ชนิดข้อมูล ชื่อตัวแปร; เช่น unsigned int nStudents;
- ตัวแปรที่อยู่ด้านซ้ายและขวาของเครื่องหมายเท่ากับจะรับคนละบทบาท
  - ตอนอยู่ด้านซ้ายจะเป็นการเปลี่ยนค่าตัวแปร (คัดลอกค่าจากคนอื่น)  
int x = 5; → คัดลอกค่าตัวเลข 5 ไปเก็บไว้ในตัวแปร x  
int y = x; → คัดลอกค่า x ไปเก็บไว้ที่ y ทำให้ y จะมีค่าเปลี่ยนตาม x ซึ่งมีค่าเท่ากับ 5 ตามตัวอย่างนี้
  - ตอนอยู่ด้านขวาจะเป็นการอ่านค่าที่เก็บไว้ (ให้คนอื่นลอกหรืออ่านค่า) เช่น ตัวอย่างเดิม int y = x; ทำคุณสมบัตินี้ให้ดูแล้ว คือ ตัวแปร x อยู่ทางขวาของเครื่องหมายเท่ากับ จะแปลว่าให้อ่านค่าปัจจุบันที่ x เก็บไว้ เมื่ออ่านได้แล้วก็จะคัดลอกค่าไปเก็บไว้ที่ y ด้วย

## สรุปกฎพื้นฐานของการใช้ตัวแปร (2)



- เราสามารถระบุค่าคงที่ให้ไปเก็บไว้ที่ตัวแปรตอนประกาศได้ เช่น  
int x = 5;  
float f = 7.0;  
double d = 7.0;  
char c = 'A'; ตัวอักขระก็จัดเป็นค่าคงที่ได้เหมือนกัน
- เราสามารถคัดลอกค่าจากตัวแปรหนึ่งไปเก็บที่ตัวแปรอีกอันได้  
ส่งผลให้ตัวแปรสองตัวมีค่าเท่ากัน
- แต่เราไม่สามารถคัดลอกค่าจากตัวแปรไปใส่ไว้ที่ค่าคงที่ได้ เพราะค่าคงที่เป็นสิ่งที่เปลี่ยนไม่ได้ มันจะไม่อยู่ด้านซ้ายของเครื่องหมายเท่ากับเป็นอันขาด เช่น 5 = x; แบบนี้ใช้ไม่ได้ 5 = 5; ก็ใช้ไม่ได้ ถึงแม้ว่าจะไม่เปลี่ยนก็ตาม

## สรุปกฎพื้นฐานของการใช้ตัวแปร (3)



- เราอ้างถึงตัวแปรที่ประกาศไว้ก่อนหน้าได้ แต่จะอ้างถึงตัวแปรก่อนการประกาศไม่ได้ ไม่ว่าจะเป็นการอ่านค่าหรือบันทึกค่าตัวแปร เช่น

```
void main() {  
    int x;  
    x = 7; → แบบนี้ใช้ได้ เพราะเราอ้างถึง x หลังประกาศตัวแปร  
}
```

```
void main() {  
    x = 7; → แบบนี้ใช้ไม่ได้ เพราะเราอ้างถึง x ก่อนการประกาศตัวแปร  
    int x;  
}
```

## ขอบเขตการมองเห็นตัวแปร (Variable Scope)



- จากกฎพื้นฐานของการใช้ตัวแปร เราจะเห็นได้ว่า 'ถึงจะมีการประกาศตัวแปรไว้ ก็ใช่ว่าจะใช้ได้เสมอไป บางทีประกาศไว้แต่ก็ใช้ว่าโปรแกรมจะมองเห็นและอ้างถึงมันได้'  
→ ตัวแปรที่มีขอบเขตที่โปรแกรมมองเห็นและที่มีกฎแน่นอนตายตัว
- เราเรียกขอบเขตที่โปรแกรมมองเห็นตัวแปรว่า *variable scope*

## กฎเกี่ยวกับ Variable Scope



- ตัวแปรที่ประกาศไว้ในวงเล็บปีกกา เรียกว่าตัวแปรเฉพาะที่ (local variable) ส่วนตัวแปรที่ประกาศไว้นอกวงเล็บปีกกาเรียกว่าตัวแปรครอบคลุม (global variable)
  - โปรแกรมจะมองเห็นตัวแปรภายหลังการประกาศแล้วเท่านั้น ไม่ว่าจะ เป็นแบบโลคอลหรือโกลบอล
  - โปรแกรมจะมองเห็นตัวแปรแบบโลคอล ภายในบริเวณวงเล็บปีกกา เดียวกัน และ วงเล็บปีกกาที่ซ้อนอยู่ข้างใน (จะซ้อนอยู่กี่ชั้นก็มองเห็นหมด) แต่ก็มองเห็นหลังการประกาศตัวแปรแล้วเท่านั้น (กฎข้อที่ 1 สำคัญที่สุด)
  - โปรแกรมจะมองเห็นตัวแปรแบบโกลบอล ภายในวงเล็บปีกกาทุกอันที่ ตามหลังการประกาศตัวแปรโกลบอลนั้น (กฎข้อที่ 1 ใหญ่คับฟ้าจริง ๆ) (ดูหนังสือหน้า 60 – 62 สำหรับตัวอย่างเพิ่มเติม)

## ตัวอย่างเรื่อง Variable Scope



```
int g1;

void main() {
    int w;
    {
        int x;
        {
            int y;
        }
        int z;
    }
}

int g2;
```

g1 และ g2 เป็นตัวแปรแบบโกลบอล เพราะการประกาศไม่อยู่ในวงเล็บปีกกา

w, x, y, และ z เป็นตัวแปรแบบโลคอล เพราะการประกาศอยู่ในวงเล็บปีกกา (จะอยู่ในวงเล็บปีกกาชั้นที่เท่าไรก็ จัดเป็นโลคอลทั้งนั้น)

สังเกตด้วยว่า main ตามด้วยวงเล็บปีกกา เครื่องหมายปีกกาตรงนี้มีมือใหม่มักจะลืม

## โปรแกรมมองเห็นตัวแปรอะไรบ้าง



```
int g1;

void main() {
    int w;
    {
        int x;
        {
            int y;
        }
        int z;
    }
}

int g2;
```

ตรงนี้โปรแกรมมองเห็นเฉพาะ g1 เท่านั้น

ตรงนี้มองเห็น g1 และ w

ตรงนี้มองเห็น g1, w และ x

ตรงนี้มองเห็น g1, w, x และ y

ตรงนี้มองเห็น g1, w, x และ z

แต่มองไม่เห็น y ถึงแม้ y จะประกาศไว้ก่อน แต่ y อยู่ในวงเล็บด้านใน

ตรงนี้มองเห็น g1 และ g2 เท่านั้น

## ความสำคัญของ variable scope



- ทำให้เราจัดเตรียมค่าที่จะใช้ได้ถูกที่ถูกลงเวลา

```
int g = 5;

void main() {
    int w = g + 2;
    {
        int x = g + w;
    }
}

void test() {
    int y = g + 9;
}
```

มีการอ้างถึง g ทั้งสองฟังก์ชัน ดังนั้น g ต้องเป็นแบบโกลบอล

## การแบ่งแยกตัวแปรและงานด้วยวงเล็บปีกกา



**คำถาม** ในตัวอย่างที่ผ่านมาเราเห็นการใช้วงเล็บปีกกาซ้อนกัน ทำแบบนี้แล้วจะมีประโยชน์อะไร

**คำตอบ** ในบางครั้ง มันช่วยให้เราจัดการโค้ดง่ายขึ้น เพราะตัวแปรที่อยู่ในวงเล็บปีกกาจะไม่ไปปะปนกับของที่อยู่ข้างนอก หากเราทำให้โค้ดในวงเล็บปีกกาทำงานเฉพาะกิจบางอย่าง การสร้างวงเล็บเพิ่มแบบนี้จะทำให้โค้ดถูกแบ่งออกเป็นสัดส่วนไม่ปะปนกัน

**Trivia :** วงเล็บปีกกาแต่ละคู่และโค้ดที่อยู่ข้างในวงเล็บนั้น มีชื่อเรียกว่า “Code Block” คาดว่าเป็นที่มาของชื่อโปรแกรมที่เราใช้ในแล็บ

## ตัวอย่างการใช้วงเล็บปีกกาแยกงาน



```
void main() {  
    int w = 5;  
  
    {  
        int x = w + 3;  
        printf(“%d\n”, x);  
    }  
  
    {  
        int x = w + 5;  
        printf(“%d\n”, x);  
    }  
}
```

บล็อกนี้เอา w มาบวก 3

บล็อกนี้เอา w มาบวก 5

โอ๊ะ ทั้งสองบล็อกมีตัวแปรชื่อ x ทั้งคู่ แบบนี้ถือว่าตั้งชื่อตัวแปรซ้ำและผิดกฎหรือไม่ ?

หมายเหตุ เพื่อความกะทัดรัด โค้ดบางส่วน เช่น #include <stdio.h> ไม่ถูกนำมาแสดง

## การซ้ำกันของชื่อตัวแปรที่อยู่ต่างบล็อก



- ถึงแม้เราจะมีกฎการตั้งชื่อตัวแปรว่าห้ามซ้ำกัน แต่แท้จริงแล้วกฎนั้นเจาะจงกับตัวแปรภายในบล็อกเดียวกันเท่านั้น
- ตัวแปรที่อยู่คนละบล็อกกันชื่อซ้ำกันได้
  - นี่เป็นเหตุผลว่าการแบ่งบล็อก (หรือฟังก์ชัน) ทำให้เราสามารถแบ่งโปรแกรมเป็นส่วน ๆ ที่อิสระจากกันได้ ทำให้เราไม่ต้องกังวลว่าการเขียนในที่หนึ่งจะไปกระทบกับอีกที่หนึ่ง
- แม้แต่บล็อกที่ซ้อนเข้าไปก็มีชื่อตัวแปรที่ซ้ำกันได้
  - ในกรณีนี้ตัวแปรที่ประกาศไว้ล่าสุดจะมีลำดับความสำคัญสูงกว่า

## ตัวอย่างชื่อตัวแปรซ้ำในบล็อกที่ซ้อนกัน



```
void main() {  
    int x = 5;  
    int w = 2;  
  
    {  
        int x = w + 1;  
        printf(“%d %d\n”, x, w);  
    }  
  
    printf(“%d %d\n”, x, w);  
}
```

มีชื่อ x ซ้ำได้ แม้ว่าจะจะเป็นบล็อกที่ซ้อนกันก็ตาม

x ตรงนี้เป็นไปตามที่ประกาศล่าสุดที่โปรแกรมมองเห็น มีค่าเท่ากับ 3

x ตรงนี้ก็จะเป็นไปตามที่ประกาศล่าสุดที่โปรแกรมมองเห็น มีค่าเท่ากับ 5

ผลลัพธ์ที่พิมพ์ออกมาคือ 3 2  
5 2

## คำถามแบบฝึกหัด (1)



ผลลัพธ์จากโปรแกรมทางด้านใต้คือเท่าใด

```
void main() {
    int x = 5;
    int w = 2;

    {
        w = x + 1;
        int x = w + 1;
        printf("%d %d\n", x, w);
        x = x - 1;
    }
    printf("%d %d\n", x, w);
}
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

57

## คำถามแบบฝึกหัด (2)



ตรงไหนบ้างที่มีการอ้างอิงถึงตัวแปรที่ผิดไป (มี 2 ตำแหน่ง)

```
int g1 = 1;
void main() {
    int x = 5;
    int w = g1 + g2;

    {
        y = x + 1;
        int y;
        int x = w + g1;
    }
}
int g2;
```

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

58

## Memory Constant



- คือค่าที่เราสั่งให้เครื่องจำไว้ และไม่ให้เปลี่ยนค่าเป็นอันขาด
  - ก่อนหน้านี้เราให้ตัวแปรเปลี่ยนค่าได้ เช่น

```
int x = w + 1;
x = x - 1;
```
  - แต่ในบางครั้งเราไม่ต้องการให้มีการเปลี่ยนค่า เพราะของบางอย่าง เช่น ค่า  $\pi$  ควรเป็นค่าคงที่ตลอด
  - เพื่อป้องกันการเปลี่ยนค่าพวกนี้โดยบังเอิญ เราจึงมีการใช้คำพิเศษ `const` เพื่อบังคับให้คอมไพเลอร์ป้องกันการแก้ค่าพวกนี้
  - เช่น `const double PI = 3.1415926535;`

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

59

## ตัวอย่างการใช้ const



```
#include <stdio.h>
void main() {
    const double PI = 3.1415926535;
    double radius = 1.5;
    double circle_area = PI * radius * radius;
    radius = radius + 1;
    PI = PI + 1;
    circle_area = PI * radius * radius;
}
```

เกิด Error ขึ้นตอนคอมไพล์  
เพราะเราระบุไว้ก่อนหน้าว่า  
PI เป็นค่าคงที่ const

- ดูตัวอย่าง 5.7 ในหนังสือเพิ่มเติม

16 กันยายน 2557

ภิญโญ แท้ประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

60

## การแสดงผลขั้นพื้นฐาน



- เรามีการพิมพ์ผลลัพธ์ออกทางจอภาพหลายครั้งแล้ว
- เราพิมพ์ออกมาได้ด้วยคำสั่ง printf
  - printf เป็นคำสั่ง สำหรับแสดงผล
  - printf ต้องการให้เราระบุข้อความ (สตริง) ที่เราต้องการแสดง
  - เราต้องนำข้อความที่ต้องการพิมพ์ไปใส่ไว้ในวงเล็บหลัง printf
- ตัวอย่าง
  - printf("Welcome to Silpakorn");
  - printf("A");
  - printf("My name is Pinyo Taeprasartsit");

16 กันยายน 2557

ภิญโญ แห่งประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

61

## การแสดงค่าจากตัวแปร



- ในกรณีที่ข้อความไม่ตายตัวแต่เปลี่ยนไปตามค่าจากตัวแปร
  - ➔ ส่งตัวแปรไปในข้อความด้วยสัญลักษณ์พิเศษ เช่น

```
int x = 5;
printf("%d", x);
```
  - สัญลักษณ์พิเศษคือ %d ซึ่งระบุว่าเราต้องการพิมพ์ค่าจำนวนเต็ม
  - เราต้องใช้ %d คู่กับตัวแปรจำนวนเต็ม ด้วยการส่งตัวแปรนั้นไปด้วยกัน
- ตัวอย่างเพิ่มเติม

```
float n = 3.0;   printf("%f", n);
char k = 'a';   printf("%c", k);
```

สัญลักษณ์พิเศษเปลี่ยนไปตามชนิดข้อมูล float ใช้ f และ char ใช้ c

16 กันยายน 2557

ภิญโญ แห่งประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

62

## การแสดงข้อความผสมตัวแปร



- เราสามารถเอาข้อความไปผสมกับตัวแปรได้เลย เช่น
  - printf("The value of x is %d", x);
  - printf("The value of n is %f", n);
- จะมีตัวแปรมากกว่าหนึ่งตัวในข้อความเดียวกันก็ได้ เช่น
  - printf("The values of x and n are %d and %f", x, n);
- ลำดับการใส่ตัวแปรไว้ด้านท้ายของ printf จะเป็นไปตามลำดับของเครื่องหมายพิเศษที่ใส่ไป
  - ตัวอย่างเพิ่มเติม

```
int x = 5;   int y = 7;
printf("%d %d", x, y); // ผลลัพธ์ทางจอภาพคือ 5 7
printf("%d %d", y, x); // ผลลัพธ์ทางจอภาพคือ 7 5
```

16 กันยายน 2557

ภิญโญ แห่งประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

63

## การแสดงข้อความผสมตัวแปรและอักขระพิเศษ



- อักขระพิเศษ เช่น ตัวขึ้นบรรทัดใหม่ (new line) \n เป็นที่นิยมใช้มาก เพราะทำให้เกิดการจัดรูปข้อความที่สวยงาม
- ตัวอย่าง : จงพิมพ์ตัวแปรที่แทนคู่ลำดับสองคู่ หนึ่งคู่ต่อบรรทัด โดยมีตัวแปรชื่อ x1, y1, x2, และ y2 ซึ่งเป็นจำนวนเต็ม ให้มีผลลัพธ์ในรูปแบบ
- ```
(x1, y1)
(x2, y2)
```
- โค้ด : แบบม้วนเดียวจบ

```
printf("(%d, %d)\n(%d, %d)", x1, y1, x2, y2);
```
- โค้ด : แบบอ่านง่าย

```
printf("(%d, %d)\n", x1, y1);
printf("(%d, %d)", x2, y2);
```

16 กันยายน 2557

ภิญโญ แห่งประสาทสิทธิ์ มหาวิทยาลัยศิลปากร

64



### คำถามแบบฝึกหัด (3)



จงหาคำตอบว่าโปรแกรมจะพิมพ์อะไรออกมาทางจอภาพ

```
int i = 1;
int j = 2;
void main() {
    int x = 3;
    int y = 4;

    printf("(%d, %d)", y, y);
    printf("%d %d\n", i, j);
    printf("%d%d%d%d", i, j, x, y);
}
```

คำแนะนำ : ระวังการเว้นวรรคและขึ้นบรรทัดใหม่ในคำตอบ

ลองทำและตรวจคำตอบด้วยการเขียนลงใน Code::Blocks

อย่าลืมแปะ #include <stdio.h> ด้วย

### คำถามแบบฝึกหัด (4)



จงหาคำตอบว่าโปรแกรมจะพิมพ์อะไรออกมาทางจอภาพ

```
void main() {
    int w = 5; int x = 3; int y = 2;
    {
        int x = w + 3;
        printf("%d\n", x);
        {
            int x = w + y;
            printf("%d\n", x);
            w = w - 1; x = x - 1;
        }
        printf("(%d, %d)", w, x);
    }
    printf("(%d, %d)", w, x);
}
```

### คำถามแบบฝึกหัด (5)



ตรงไหนบ้างที่มีการอ้างถึงตัวแปรที่ผิดไป (มี 3 ตำแหน่ง หาให้ครบด้วย)

```
int g1 = 1;
void main() {
    int x = 5;
    int w = g1 + g2;

    {
        y = x + 1;
        int y;
        int x = w + g1;
    }
    printf("%d %d\n", x, y);
}
int g2;
```

### เรื่องอื่น ๆ



- ทำแบบฝึกหัดท้ายบทที่ 5 ในหนังสือเรียนข้อ 1 – 4 ด้วย  
มีเฉลยอยู่ด้านหลังท้ายของเล่ม ทำแล้วตรวจคำตอบได้เลย
- เราไม่เรียนหัวข้อต่อไปนี้ในบทที่ 5
  - Storage class (พวก static, auto, extern)
  - ไม่เรียนเรื่อง Defined Constant  
(แต่เรียนเรื่อง Memory Constant)