Computer Programming I:
การเขียนโปรแกรมคอมพิวเตอร์ I

ฟังก์ชันไลบรารี่มาตรฐาน (Standard Library Functions) และ สตริง (String)

อ.ดร.ปัญญนัท อ้นพงษ์

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

aonpong_p@su.ac.th

Outline

- **ฟังก์ชันไลบรารี่มาตรฐาน**
  - **นิยาม**
  - Header File
  - ฟังก์ชันคณิตศาสตร์

- สายอักขระ (สตริง; String)
  - ข้อมูลชนิดอักขระ (character)
  - ฟังก์ชันที่เกี่ยวข้องกับ character
  - สตริง (String)
  - การจัดการ String

ฟังก์ชันไลบรารี่มาตรฐาน: นิยาม

- ก่อนหน้านี้เราได้ทดลองสร้างฟังก์ชันด้วยตนเองแล้ว
- จะพบว่าการเขียนฟังก์ชันเพียงครั้งเดียว ทำให้เราสามารถเรียกฟังก์ชันได้ซ้ำๆ
- หลายครั้งที่โปรแกรมเมอร์เขียนฟังก์ชันที่ตัวเองใช้บ่อยๆ เอาไว้ เพื่อให้สามารถเรียกใช้ได้ทันที ไม่ต้องเขียนโปรแกรมใหม่
- แต่ในความเป็นจริง ก็จะมีฟังก์ชันคณิตศาสตร์มากมายที่ไม่ถึงกับซับซ้อนมาก แต่ภาษาซีกลับไม่รองรับ ทำให้เมื่อโปรแกรมเมอร์จะเรียกฟังก์ชันเหล่านี้ จำเป็นต้องเขียนฟังก์ชันเอง
- แต่ด้วยความเอื้ออาทรของโปรแกรมเมอร์กลุ่มหนึ่ง เขาได้ทำการเขียนฟังก์ชันที่เราอาจต้องใช้บ่อยๆ นี้ไว้ให้พวกเราแล้ว (เราก็แค่เรียกใช้อย่างเดียว)

ฟังก์ชันไลบรารี่มาตรฐาน: นิยาม

- ฟังก์ชันทั้งมวลที่ต้องใช้บ่อยๆ จะถูกรวบรวมและเก็บเป็นคลัง เรียกว่า Library ดังนั้น Library ในที่นี้จึงหมายถึง ศูนย์รวมคำสั่ง
- และ Standard Library Function ก็คือ ศูนย์รวมฟังก์ชันมาตรฐานนั่นเอง
  - SLF จะพบได้ในคอมไพล์เลอร์ภาษาซีที่ได้มาตรฐานทุกตัว
  - ดังนั้นเราสามารถเปลี่ยนคอมไพล์เลอร์ที่ใช้ไปเป็นตัวอื่นได้ โค้ดเดิมที่เรียกใช้ SLF ก็จะยังคงใช้งานได้เหมือนเดิม
  - ตัวอย่างฟังก์ชันมาตรฐานที่เราใช้มาตลอด คือ printf และ scanf ที่อยู่ใน Library ชื่อ stdio.h

ฟังก์ชันไลบรารี่มาตรฐาน: นิยาม

- ในชีวิตจริง เราอาจต้องพึ่งพาไลบรารี่ที่ไม่ใช่มาตรฐานด้วย เพราะไลบรารี่มาตรฐานจะไม่ครอบคลุมการทำงานทุกรูปแบบ
  - บางบริษัทมีการทำงานแบบเฉพาะทาง เขาอาจพัฒนาไลบรารี่ของตัวเองขึ้นมาใช้
  - บางหัวข้อการศึกษาที่เป็นที่สนใจในสังคมนักวิจัย ก็จะมีนักพัฒนาพยายามทำไลบรารี่ใหม่ๆ มาช่วยนักวิจัยด้วย แต่การใช้งานก็จะแพร่หลายแค่ในวงการเท่านั้น
- ฟังก์ชันไลบรารี่มาตรฐาน แม้จะไม่ครอบคลุมทุกอย่าง แต่ก็มีข้อดี
  - จะเปลี่ยน platform ก็ไม่ต้องเขียนโปรแกรมใหม่ (SLF รองรับอยู่แล้วทุกคอมไพเลอร์)
  - ภาษาโปรแกรมเมอร์จะเรียกความสามารถในการย้ายสภาพแวดล้อมแบบนี้ว่า “portability”

Outline

- **ฟังก์ชันไลบรารี่มาตรฐาน**
  - นิยาม
  - Header File
  - ฟังก์ชันคณิตศาสตร์

- สายอักขระ (สตริง; String)
  - ข้อมูลชนิดอักขระ (character)
  - ฟังก์ชันที่เกี่ยวข้องกับ character
  - สตริง (String)
  - การจัดการ String

ฟังก์ชันไลบรารี่มาตรฐาน: Header File

- เรากำลังพูดถึงบรรทัดแรกของการเขียนทุกโปรแกรมที่เราเคยเขียนมา

```
#include <stdio.h>
```

- ในบรรทัดนี้คือการเรียกไลบรารี่มาตรฐานโดยผ่าน header file (ลงท้ายด้วย *.h หรืออาจเป็นตัวอื่นก็ได้)
- เมื่อเรานำเข้า header file โปรแกรมของเราจะสามารถเรียกใช้สิ่งที่บรรจุอยู่ภายใน header file นั้นได้ ทั้งฟังก์ชันและตัวแปรบางอย่าง
  - เช่นถ้าเรานำเข้า stdio.h เราจะสามารถ printf และ scanf ได้ ถ้าไม่นำเข้า คอมไพล์เลอร์จะแจ้งว่าหาฟังก์ชัน printf และ scanf ไม่เจอ (ไปลองทำดูก็ได้)

ฟังก์ชันไลบรารี่มาตรฐาน: Header File

- ไลบรารี่มาตรฐานถูกแบ่งหมวดหมู่ตาม Header file ตัวที่สำคัญและมีโอกาสใช้บ่อยในชั้นเรียนมีดังนี้

Header file	ข้อมูลเบื้องต้น
math.h	รวบรวมคำสั่งเกี่ยวกับการคำนวณทางคณิตศาสตร์
limits.h	รวบรวมค่าคงที่สำคัญ (INT_MAX, INT_MIN, etc.)
stdlib.h	เครื่องมืออรรถประโยชน์
ctype.h	เครื่องมือช่วยเหลือสำหรับอักขระ
string.h	เครื่องมือช่วยเหลือสำหรับสายอักขระ

Outline

- **ฟังก์ชันไลบรารี่มาตรฐาน**
  - นิยาม
  - Header File
  - **ฟังก์ชันคณิตศาสตร์**

- สายอักขระ (สตริง; String)
  - ข้อมูลชนิดอักขระ (character)
  - ฟังก์ชันที่เกี่ยวข้องกับ character
  - สตริง (String)
  - การจัดการ String

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

#include<math.h> เพื่อใช้ฟังก์ชันต่อไปนี้ (เซ็ตที่ 1: หมวดพื้นฐาน)

Function	แม่แบบ	ข้อมูลเบื้องต้น
sqrt	double **sqrt** (double input);	ใช้หารากที่ 2 ของข้อมูลที่เป็น input
pow	double **pow** (double base, double exp);	ใช้หาผลยกกำลัง โดยจะได้ค่า $base^{exp}$
exp	double **exp** (double arg);	ใช้หาผลยกกำลังฐาน e (ค่า $e \approx 2.718$) โดยจะได้ค่า e^{arg}
log	double **log** (double arg);	ใช้หา log โดยใช้ฐาน e โดยจะได้ค่า $\ln(\text{arg})$

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

การใช้งานฟังก์ชันมาตรฐาน

1. include ไลบรารี่ที่เราต้องการใช้ เช่น #include <math.h>
2. เรียกใช้ฟังก์ชันที่ต้องการ โดยอ่านจากแม่แบบฟังก์ชัน

- เช่นถ้าต้องการเรียกใช้ sqrt ก็ต้องพิจารณาแม่แบบฟังก์ชันต่อไปนี้

  double **sqrt** (double input);

  - จากฟังก์ชันนี้ return type คือ double ดังนั้นตัวแปรที่จะมาเก็บคำตอบของฟังก์ชันนี้จะต้องเป็นตัวแปรประเภท double
  - ชื่อฟังก์ชันคือ sqrt ดังนั้นเวลาจะเรียกใช้ฟังก์ชันนี้ก็ต้องใช้คีย์เวิร์ดว่า sqrt( )
  - ข้อมูลที่ต้องใส่เข้าไปคือตัวเลขชนิด double ดังนั้น ค่าที่ต้องใส่ในวงเล็บหลัง sqrt ก็ต้องเป็นตัวเลข หรือตัวแปรประเภท double

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

```c
#include<stdio.h>
#include<math.h>
 void main(){
      double input = 16;
      double result;
      result = sqrt(input);
      printf("%lf", result);
 }
```

```
4.000000
```

```c
#include<stdio.h>
#include<math.h>
 void main(){
      double x;
      scanf("%lf", &x);
      double result;
      result = sqrt(x);
      printf("%lf", result);
 }
```

```
32
5.656854
```

```c
#include<stdio.h>
#include<math.h>
 void main(){
      printf("%lf", sqrt(64));
 }
```

```
8.000000
```

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโจทย์ (1) ประยุกต์ใช้ฟังก์ชัน sqrt

จงเขียนโปรแกรมที่รับความยาวด้านประกอบมุมฉากทั้งสอง เมื่อความยาวด้านประกอบมุมฉากทั้งสองเป็นจำนวนจริงบวก จากนั้นคำนวณหาความยาวด้านตรงข้ามมุมฉาก พร้อมแสดงผลออกทางจอภาพ

วิเคราะห์ จำนวนจริงหมายความรวมถึงเลขจำนวนเต็มและทศนิยม

ข้อมูลขาเข้า ความยาวด้านประกอบมุมฉากทั้งสอง (a และ b)

ข้อมูลขาออก ความยาวด้านตรงข้ามมุมฉาก (c) โดยคำนวณจากสมการ $c^2 = a^2 + b^2$, และเมื่อแก้สมการจะได้ว่า $c = \sqrt{a^2 + b^2}$

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

```c
#include <stdio.h>
#include <math.h>
void main() {
    double a, b;
    scanf("%lf %lf", &a, &b);
    double result = sqrt(a*a + b*b);
    printf("%lf", result);
}
```

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโจทย์ (2) ประยุกต์ใช้ฟังก์ชัน pow

จงเขียนโปรแกรมที่รับความยาวด้านประกอบมุมฉากทั้งสอง เมื่อความยาวด้านประกอบมุมฉากทั้งสองเป็นจำนวนจริงบวก จากนั้นคำนวณหาความยาวด้านตรงข้ามมุมฉาก พร้อมแสดงผลออกทางจอภาพ โดยใช้ฟังก์ชัน pow

วิเคราะห์โจทย์เพิ่มเติม ฟังก์ชันแม่แบบของ pow คือ double **pow** (double base, double exp);

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

```c
#include <stdio.h>
#include <math.h>
void main() {
    double a, b;
    scanf("%lf %lf", &a, &b);
    double result = sqrt(pow(a,2) + pow(b,2));
    printf("%lf", result);
}
```

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโจทย์ (3) ประยุกต์ใช้ฟังก์ชัน pow แบบประยุกต์จริง

จงเขียนโปรแกรมหาค่า x เมื่อกำหนดให้ $y = x^3$ โดยที่ y คือค่าที่รับจากผู้ใช้

วิเคราะห์โจทย์ ถ้า $y = x^3$ แล้ว $x = \sqrt[3]{y}$ และ $\sqrt[3]{y} = y^{\frac{1}{3}}$

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโจทย์ (3) ประยุกต์ใช้ฟังก์ชัน pow แบบประยุกต์จริง

จงเขียนโปรแกรมหาค่า x เมื่อกำหนดให้ $y = x^3$ โดยที่ y คือค่าที่รับจากผู้ใช้

วิเคราะห์โจทย์ ถ้า $y = x^3$ แล้ว $x = \sqrt[3]{y}$ และ $\sqrt[3]{y} = y^{\frac{1}{3}}$

```
#include <stdio.h>
#include <math.h>
void main() {
    double x, y;
    scanf("%lf", &y);
    x = pow(y, 1.0/3.0);
    printf("%lf", x);
}
```

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

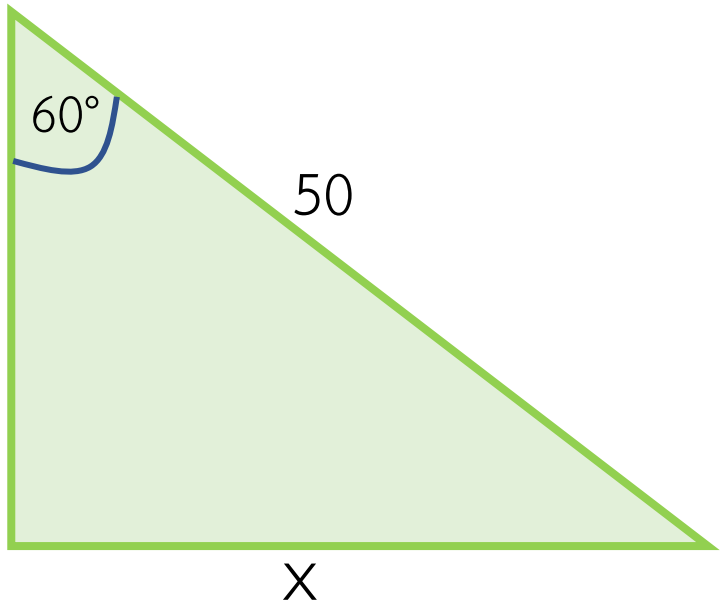
#include<math.h> เพื่อใช้ฟังก์ชันต่อไปนี้ (เซ็ตที่ 2: หมวดตรีโกณมิติ)

Function	แม่แบบ	ข้อมูลเบื้องต้น
sin	double sin (double arg);	หาค่า sin arg โดย arg เป็นมุม มีหน่วย rad
cos	double cos (double arg);	หาค่า cos arg โดย arg เป็นมุม มีหน่วย rad
tan	double tan (double arg);	หาค่า tan arg โดย arg เป็นมุม มีหน่วย rad
asin	double asin (double arg);	หา arcsin arg โดย arg เป็นมุม มีหน่วย rad
acos	double acos (double arg);	หา arccos arg โดย arg เป็นมุม มีหน่วย rad
atan	double atan (double arg);	หา arctan arg แบบ 1 quadrant
atan2	double atan2 (double y, double x);	หา arcsin arg แบบ 2 quadrant (ใช้เครื่องหมายของพารามิเตอร์ในการหา quadrant)

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโจทย์ (4) ประยุกต์ใช้ฟังก์ชันตรีโกณมิติ

จงหาความยาวของด้านประกอบมุมฉาก x เมื่อกำหนดความยาวของด้านตรงข้ามมุมฉาก และขนาดของมุมดังภาพ

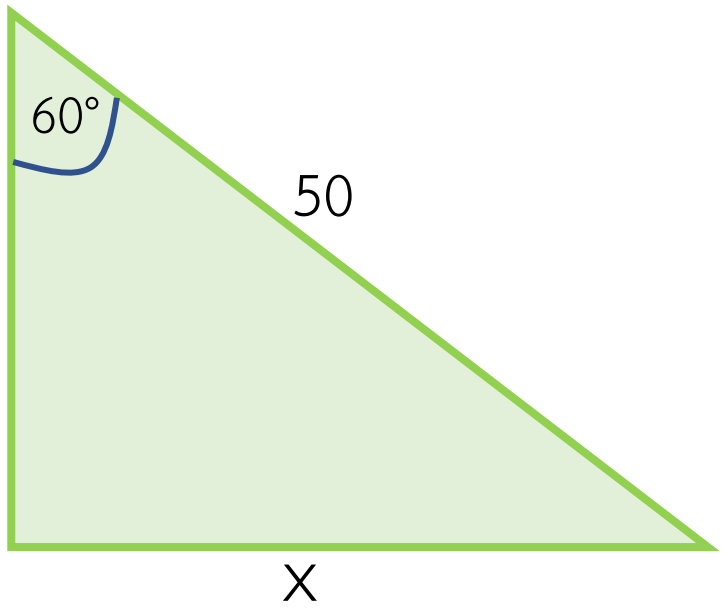


Hint: $radian = \frac{degree}{180}\pi$

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโจทย์ (4) ประยุกต์ใช้ฟังก์ชันตรีโกณมิติ

จงหาความยาวของด้านประกอบมุมฉาก x เมื่อกำหนดความยาวของด้านตรงข้ามมุมฉาก และขนาดของมุมดังภาพ



วิธีคิด เราต้องคำนวณ 50 sin(60 °)

แต่ถ้าจะใช้ไลบรารี่ เราต้องใช้มุม radian

ต้องมีการแปลงมุมด้วยสมการ $radian = \frac{degree}{180}\pi$

จะคิดเองก็ได้ หรือจะเขียนเพิ่มในโปรแกรมก็ได้

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโจทย์ (4) ประยุกต์ใช้ฟังก์ชันตรีโกณมิติ

```c
#include <stdio.h>
#include <math.h>
void main() {
    double angle = (60.0 / 180.0) * M_PI;
    printf("%lf", 50 * sin(angle));
}
```

M_PI คือค่า π ที่มีความแม่นยำสูง จากไลบรารี่ math.h

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

การประยุกต์ที่ใช้ในงานคณิตศาสตร์ในระดับที่สูงขึ้น (ดูให้พอรู้)

เราสามารถหา Dot product ของเวคเตอร์ 2 เวคเตอร์ได้จากสมการ

$$\vec{u} \cdot \vec{v} = \|\vec{u}\|\|\vec{v}\| \cos\theta$$

- เป็นสมการที่พบได้ทั่วไปในงานทางวิทยาศาสตร์ โดยเฉพาะสายฟิสิกส์
- ถ้าอยากจะรู้มุมที่ทั้งสองเวคเตอร์ทำต่อกัน ก็จะต้องปรับสูตรย้ายข้างนิดหน่อย จะได้ว่า

$$\cos\theta = \frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|}$$

$$\theta = \arccos\left(\frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|}\right)$$

- ก็ต้องกดเครื่องคิดเลขกันยาวๆ ในทุกครั้งที่ต้องการคำนวณ หรือไม่ก็เขียนโปรแกรมครั้งเดียวไปเลย

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโจทย์ (5) ประยุกต์ใช้ในงานคำนวณที่สูงขึ้น

กำหนดเวคเตอร์ $\vec{u} = \begin{bmatrix} 3 \\ 2 \end{bmatrix}$ และ $\vec{v} = \begin{bmatrix} 1 \\ 4 \end{bmatrix}$ จงเขียนโปรแกรมเพื่อหามุมระหว่างเวคเตอร์ทั้งสองนี้

วิเคราะห์ เราต้องการคำนวณ $\theta = \arccos\left(\frac{\vec{u}\cdot\vec{v}}{\|\vec{u}\|\|\vec{v}\|}\right)$

1. คำนวณ dot product ได้จาก $\vec{u} \cdot \vec{v}$ = (3 x 1) + (2 x 4)
2. ขนาดของเวคเตอร์ทั้งสองหาได้จาก $\|\vec{u}\| = \sqrt{3^2 + 2^2}$ และ $\|\vec{v}\| = \sqrt{1^2 + 4^2}$

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโจทย์ (5) ประยุกต์ใช้ในงานคำนวณที่สูงขึ้น

```c
#include <stdio.h>
#include <math.h>
void main() {
    double x1 = 3, y1 = 2, x2 = 1, y2 = 4;
    double dot = x1*x2 + y1*y2;
    double size1 = sqrt(x1*x1 + y1*y1);
    double size2 = sqrt(x2*x2 + y2*y2);
    double theta =
    acos(dot / (size1*size2));
    printf("Angle = %lf", theta);
}
```

ตอนนี้ theta เป็นหน่วย radian

ถ้าต้องการให้ theta เป็นหน่วยองศา จะต้องทำอย่างไร

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

#include<math.h> เพื่อใช้ฟังก์ชันต่อไปนี้ (เซ็ตที่ 3: หมวดจัดการทศนิยม)

Function	แม่แบบ	ข้อมูลเบื้องต้น
ceil	double **ceil** (double x);	ปัดทศนิยมของ x **ขึ้น**ให้เป็นจำนวนเต็ม
floor	double **floor** (double x);	ปัดทศนิยมของ x **ลง**ให้เป็นจำนวนเต็ม

เช่น ถ้า x = 1.23 จะได้ผลของ ceil(x) เป็น 2.00 และจะได้ผลของ floor(x) เป็น 1.00

ถ้า x = -1.23 จะได้ผลของ ceil(x) เป็น -1.00 (เพราะว่า -1.00 มีค่ามากกว่า -2.00)

และจะได้ผลของ floor(x) เป็น -2.00 (เพราะว่า -2.00 มีค่าน้อยกว่า -1.00)

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

ตัวอย่างโค้ดที่ใช้ฟังก์ชัน ceil และ floor

```c
#include <stdio.h>
#include <math.h>
void main() {
    double x = 1.23;
    double y = -1.23;
    double ceil_x = ceil( x );
    double ceil_y = ceil( y );
    printf("%lf %lf\n", ceil_x, ceil_y);
    double floor_x = floor( x );
    double floor_y = floor( y );
    printf("%lf %lf\n", floor_x, floor_y);
}
```

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

สรุปฟังก์ชันคณิตศาสตร์ (1)

Function	แม่แบบ	ข้อมูลเบื้องต้น
sqrt	double **sqrt** (double input);	ใช้หารากที่ 2 ของข้อมูลที่เป็น input
pow	double **pow** (double base, double exp);	ใช้หาผลยกกำลัง โดยจะได้ค่า $base^{exp}$
exp	double **exp** (double arg);	ใช้หาผลยกกำลังฐาน e (ค่า $e \approx 2.718$) โดยจะได้ค่า e^{arg}
log	double **log** (double arg);	ใช้หา log โดยใช้ฐาน e โดยจะได้ค่า $\ln(\text{arg})$

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

สรุปฟังก์ชันคณิตศาสตร์ (2)

Function	แม่แบบ	ข้อมูลเบื้องต้น
sin	double sin (double arg);	หาค่า sin arg โดย arg เป็นมุม มีหน่วย rad
cos	double cos (double arg);	หาค่า cos arg โดย arg เป็นมุม มีหน่วย rad
tan	double tan (double arg);	หาค่า tan arg โดย arg เป็นมุม มีหน่วย rad
asin	double asin (double arg);	หา arcsin arg โดย arg เป็นมุม มีหน่วย rad
acos	double acos (double arg);	หา arccos arg โดย arg เป็นมุม มีหน่วย rad
atan	double atan (double arg);	หา arctan arg แบบ 1 quadrant
atan2	double atan2 (double y, double x);	หา arcsin arg แบบ 2 quadrant (ใช้เครื่องหมายของพารามิเตอร์ในการหา quadrant)

ฟังก์ชันไลบรารี่มาตรฐาน: math.h

สรุปฟังก์ชันคณิตศาสตร์ (3)

Function	แม่แบบ	ข้อมูลเบื้องต้น
ceil	double **ceil** (double x);	ปัดทศนิยมของ x **ขึ้น**ให้เป็นจำนวนเต็ม
floor	double **floor** (double x);	ปัดทศนิยมของ x **ลง**ให้เป็นจำนวนเต็ม

Outline

- ฟังก์ชันไลบรารี่มาตรฐาน
  - นิยาม
  - Header File
  - ฟังก์ชันคณิตศาสตร์

- **สายอักขระ (สตริง; String)**
  - **ข้อมูลชนิดอักขระ (character)**
  - ฟังก์ชันที่เกี่ยวข้องกับ character
  - สตริง (String)
  - การจัดการ String

ข้อมูลชนิด character

- ที่ผ่านมาเราเน้นการทำงานกับข้อมูลชนิดจำนวนเต็มกับจำนวนจริง (มีแต่ตัวเลข)
- ข้อมูลชนิดตัวอักขระ (character) ก็มีความสำคัญ แต่ยังไม่ได้พูดถึงมากนัก
  - เป็นพื้นฐานของการเก็บข้อมูลจำพวกชื่อและข้อความต่าง ๆ
  - เป็นพื้นฐานของการแสดงผลลัพธ์หลาย ๆ อย่าง
- คำว่าตัวอักขระในที่นี้หมายถึงตัวอักษรโดด
  - ไม่ใช่ข้อความ แต่เป็นเป็นตัวอักษรแค่**ตัวเดียว** เช่น ‘a’ หรือ ‘b’
  - หมายความรวมถึงเครื่องหมายวรรคตอน และ ตัวเลข เช่น ‘1’ หรือ ‘ ’ *(1 ไม่เท่ากับ ‘1’)*
  - รวมถึงตัวอักษรพิเศษที่มองไม่เห็น เช่น ช่องว่าง (space), ตัวขึ้นบรรทัดใหม่ (new line), หรือแม้กระทั่งเสียงเตือน (bell)

ข้อมูลชนิด character

การใช้งาน

- ถ้ายังจำกันได้ การประกาศตัวแปรชนิด character จะใช้ keyword ว่า char

```
char c;          //ตรงนี้ก็เหมือน int x; หรือ float y; หรือ double z;
```

- แต่การกำหนดค่าให้ตัวแปรจะแตกต่างจากตัวแปรประเภทตัวเลข
  - ถ้าเป็นตัวแปรประเภทตัวเลข จะกำหนดได้ตรงๆ เช่น a=5;
  - แต่ตัวแปรประเภท char จะต้องใส่เครื่องหมายอัญประกาศเดี่ยว ครอบค่าที่ต้องการไว้ด้วย (เพื่อให้ค่านั้นไม่เหมือนตัวแปร) เช่น c = ‘a’ หมายความว่าให้เอาตัว ‘a’ เก็บไว้ในตัวแปร c
  - จะไม่เหมือนกับ c = a เพราะตัวนี้จะหมายความว่าให้เอา **ค่า** ของตัวแปร a เก็บไว้ในตัวแปร c

ข้อมูลชนิด character

รหัสอักขระ

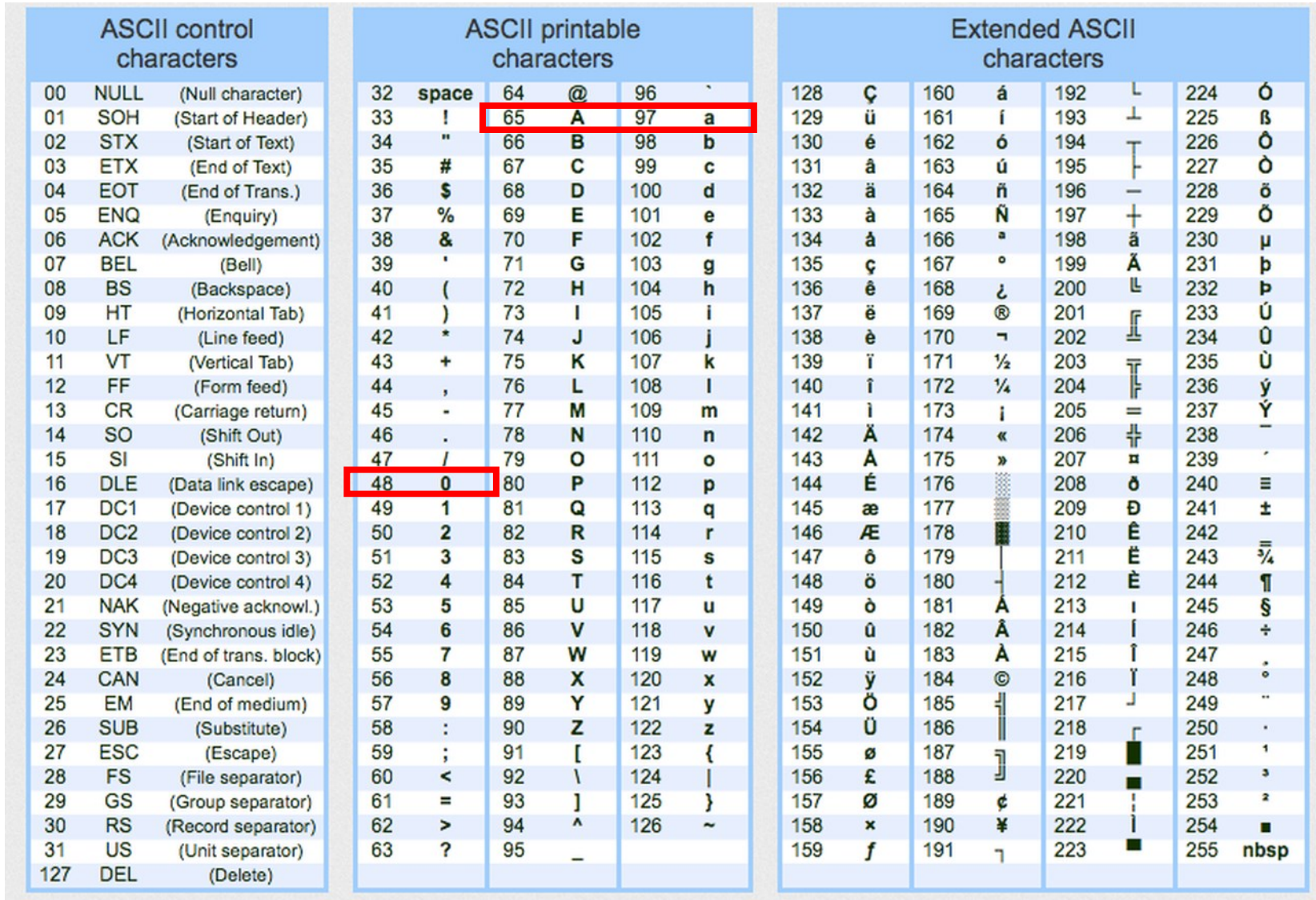
- จริงๆ แล้วคอมพิวเตอร์จำค่าได้แค่ตัวเลขเท่านั้น
- ตัวอักขระแท้จริงแล้ว คอมพิวเตอร์จดจำในลักษณะของ “ตัวเลข”
  - เช่นบอกว่า ถ้าเจอตัวแปร char ที่มีค่าเป็น 65 ให้แสดงตัว ‘A’ ออกมา เป็นต้น
- คอมพิวเตอร์จะเปลี่ยนรหัสเป็นตัวอักษรเมื่อจะแสดงผลเท่านั้น (ตอนเก็บค่าเป็นตัวเลขทั้งหมด)
  - เป็นการเชื่อมต่อกันระหว่างรูปแบบที่เครื่องเข้าใจกับรูปแบบที่มนุษย์เข้าใจ
    - เครื่องเข้าใจแต่ตัวเลข (เก็บข้อมูลทุกอย่างเป็นเลขฐานสองอยู่ภายใน)
    - มนุษย์ใช้การอ่านข้อความหรือตัวอักษร ไม่ได้ถนัดการอ่านตัวเลข
  - เราจึงให้เครื่องเก็บตัวเลขไว้ แต่ตอนแสดงผลให้มันแปลงตัวเลขเป็นตัวอักษรให้เราอ่าน
- เพื่อให้การเก็บรหัสตัวเลขของอักขระเป็นสิ่งที่เข้าใจตรงกันจึงได้มีการกำหนดมาตรฐานขึ้นมา หนึ่งในมาตรฐานที่ใช้กันอย่างแพร่หลายคือ ASCII

ข้อมูลชนิด character

รหัสอักขระ มาตรฐาน ASCII

- ย่อมาจาก American Standard Code for Information Interchange
- อักขระแต่ละตัวจะมีตัวเลขประจำอยู่ เช่น
  - A มีค่าตัวเลขเท่ากับ 65 และ a มีค่าตัวเลขเท่ากับ 97
  - ตัวเล็กกับตัวใหญ่ถือว่าเป็นคนละตัวกัน
- มีอักขระพิเศษประกอบอยู่ใน ASCII Code เช่น
  - ‘\0’ มีค่าตัวเลขเท่ากับ 0 ใช้สำหรับแทนจุดสิ้นสุดของข้อความ (สตริง)
    - เรานิยมเรียกอักขระตัวนี้ว่า null character (อักขระศูนย์) (เดี๋ยวจะถูกพูดต่อในช่วง string)
  - ‘\t’ มีค่าตัวเลขเท่ากับ 9 ใช้สำหรับแทนจุดตั้งระยะก้นหน้า (tab)

ข้อมูลชนิด character



ASCII control characters		
00	NULL	(Null character)
01	SOH	(Start of Header)
02	STX	(Start of Text)
03	ETX	(End of Text)
04	EOT	(End of Trans.)
05	ENQ	(Enquiry)
06	ACK	(Acknowledgement)
07	BEL	(Bell)
08	BS	(Backspace)
09	HT	(Horizontal Tab)
10	LF	(Line feed)
11	VT	(Vertical Tab)
12	FF	(Form feed)
13	CR	(Carriage return)
14	SO	(Shift Out)
15	SI	(Shift In)
16	DLE	(Data link escape)
17	DC1	(Device control 1)
18	DC2	(Device control 2)
19	DC3	(Device control 3)
20	DC4	(Device control 4)
21	NAK	(Negative acknowl.)
22	SYN	(Synchronous idle)
23	ETB	(End of trans. block)
24	CAN	(Cancel)
25	EM	(End of medium)
26	SUB	(Substitute)
27	ESC	(Escape)
28	FS	(File separator)
29	GS	(Group separator)
30	RS	(Record separator)
31	US	(Unit separator)
127	DEL	(Delete)

ASCII printable characters					
32	space	64	@	96	`
33	!	65	A	97	a
34	"	66	B	98	b
35	#	67	C	99	c
36	$	68	D	100	d
37	%	69	E	101	e
38	&	70	F	102	f
39	'	71	G	103	g
40	(	72	H	104	h
41	)	73	I	105	i
42	*	74	J	106	j
43	+	75	K	107	k
44	,	76	L	108	l
45	-	77	M	109	m
46	.	78	N	110	n
47	/	79	O	111	o
48	0	80	P	112	p
49	1	81	Q	113	q
50	2	82	R	114	r
51	3	83	S	115	s
52	4	84	T	116	t
53	5	85	U	117	u
54	6	86	V	118	v
55	7	87	W	119	w
56	8	88	X	120	x
57	9	89	Y	121	y
58	:	90	Z	122	z
59	;	91	[	123	{
60	<	92	\	124	|
61	=	93	]	125	}
62	>	94	^	126	~
63	?	95	_		

Extended ASCII characters							
128	Ç	160	á	192	└	224	Ó
129	ü	161	í	193	┴	225	ß
130	é	162	ó	194	┬	226	Ô
131	â	163	ú	195	├	227	Ò
132	ä	164	ñ	196	─	228	õ
133	à	165	Ñ	197	┼	229	Õ
134	å	166	ª	198	ã	230	µ
135	ç	167	º	199	Ã	231	þ
136	ê	168	¿	200	╚	232	Þ
137	ë	169	®	201	╔	233	Ú
138	è	170	¬	202	╩	234	Û
139	ï	171	½	203	╦	235	Ù
140	î	172	¼	204	╠	236	ý
141	ì	173	¡	205	═	237	Ý
142	Ä	174	«	206	╬	238	¯
143	Å	175	»	207	¤	239	´
144	É	176	░	208	ð	240	≡
145	æ	177	▒	209	Ð	241	±
146	Æ	178	▓	210	Ê	242	‗
147	ô	179	│	211	Ë	243	¾
148	ö	180	┤	212	È	244	¶
149	ò	181	Á	213	ı	245	§
150	û	182	Â	214	Í	246	÷
151	ù	183	À	215	Î	247	¸
152	ÿ	184	©	216	Ï	248	°
153	Ö	185	╣	217	┘	249	¨
154	Ü	186	║	218	┌	250	·
155	ø	187	╗	219	█	251	¹
156	£	188	╝	220	▄	252	³
157	Ø	189	¢	221	¦	253	²
158	×	190	¥	222	Ì	254	■
159	ƒ	191	┐	223	▀	255	nbsp

ข้อมูลชนิด character

การดำเนินการทาง “ตัวเลข” กับตัวแปรประเภท “อักขระ”

- จริงๆ แล้วตัวอักขระถูกคอมพิวเตอร์จดจำในฐานะตัวเลข
  - เราสามารถดำเนินการคำนวณทางตัวเลขกับมันได้
  - บวกลบค่าอักขระได้
  - เปรียบเทียบความมากน้อยของตัวเลขต่าง ๆ ได้

ข้อมูลชนิด character

ตัวอย่างโจทย์ (6) เปลี่ยนอักษรตัวพิมพ์ใหญ่ให้กลายเป็นพิมพ์เล็ก

วิเคราะห์ อักษรตัวพิมพ์ใหญ่มีรหัสน้อยกว่าตัวพิมพ์เล็กอยู่ 32 (ควรจำไว้)

65	A	97	a
66	B	98	b
67	C	99	c
68	D	100	d
69	E	101	e
70	F	102	f
71	G	103	g

นอกจากนี้ยังรู้ด้วยว่าตัวพิมพ์ใหญ่จะมีค่าอยู่ระหว่าง 65 ไปถึง 90 (65+26-1)

เราจึงสามารถตรวจสอบว่าอักขระตัวนั้นเป็นตัวพิมพ์ใหญ่หรือไม่ ด้วยประโยค

```
if ( c >= 65 && c <= 90 )
```

ข้อมูลชนิด character

ตัวอย่างโจทย์ (6) เปลี่ยนอักษรตัวพิมพ์ใหญ่ให้กลายเป็นพิมพ์เล็ก

```c
#include<stdio.h>
void main() {
    char c = 'A';
    if( c >= 65 && c <= 90 ) {
        c += 32;
    }
    printf("%c", c);
}
```

ตรวจสอบว่าเป็นตัวพิมพ์ใหญ่หรือไม่ ถ้าไม่ใช่จะไม่เปลี่ยนค่า

ถ้าเป็นพิมพ์ใหญ่ จะเพิ่มรหัสไปอีก 32 ทำให้ตัวแปร c เปลี่ยนรหัสไปเป็นตัวพิมพ์เล็กแทน (จาก ‘A’ เป็น ‘a’)

สังเกตว่า พิมพ์ตัวแปร c ไม่ใช่อักขระ ‘c’ ถ้าเปลี่ยนตรงนี้เป็น ‘c’ จะเป็นอย่างไร

ข้อมูลชนิด character

การรับค่าอักขระจากผู้ใช้

- เราใช้คำสั่ง scanf ได้เหมือนตัวแปรทั่วไป เพียงแต่ใช้ %c สำหรับการรับข้อมูล char

```
char c;

scanf("%c", &c);
```

สังเกตว่า เรารับค่ามาใส่ที่อยู่ของตัวแปร c ไม่ใช่รับค่ามาใส่ 'c' (อักขระ 'c') ดังนั้นจะต้องไม่มีเครื่องหมาย ' '

ข้อมูลชนิด character

Tip and Trick

- บนวินโดวส์เวลาที่เรากด Enter จะมีอักขระขึ้นบรรทัดใหม่กับขึ้นต้นบรรทัดใหม่แยกออกจากกัน ทำให้มีตัวอักษรเกินมาตัวหนึ่ง (ถ้าโปรแกรมรับอินพุตมาตัวเดียว ปัญหาตัวอักษรเกินจะไม่มีผลใดๆ)

```c
#include<stdio.h>
void main() {
    char c1, c2, c3;
    scanf("%c", &c1);
    scanf("%c", &c2);
    scanf("%c", &c3);

    printf("Output: %c %c %c", c1, c2, c3);
}
```

```
A
B
Output: A
 B
Process returned 13 (0xD)
Press any key to continue.
```

ข้อมูลชนิด character

- ตัวอักษรที่เกินมาจากการกด enter นี้จะถูกโอนไปใส่ตัวแปรถัดไป
  - ทำให้ถ้าเรารับตัวแปรมากกว่า 1 ตัว ตัวแปรที่รับตัวที่ 2 จะเกิดปัญหา และการแก้ไขทั่ว ๆ ไป ชวนสับสนพอสมควร
- ถ้าไม่จำเป็นจริง ๆ ควรเลี่ยงการรับข้อมูลเข้าจากผู้ใช้ในรูปแบบตัวอักษร
- ควรใช้ข้อมูลชนิดข้อความ (string) แทนตัวอักขระ แล้วจึงแยกตัวอักษรออกมาเพิ่มเติมในภายหลัง (ซึ่งเดี๋ยวจะกล่าวถึง string ต่อไป)
  - เพื่อป้องกันความสับสนกับความแตกต่างนี้ เราสามารถใช้สตริงมาเก็บข้อมูลตัวอักขระทั่วไปก็ได้
  - ตัวแปรชนิดอักขระยังมีความสำคัญในการคำนวณและแสดงผลตามปรกติ
- ไม่ใช่ทุกภาษา คอมไพเลอร์ หรือทุกระบบปฏิบัติการจะเป็นแบบนี้

ข้อมูลชนิด character

Tip and Trick ส่วนตัว (เผื่อเจอปัญหานี้ตอนทำงาน/ไม่การันตีว่าใช้ได้ทุก platform)

- จริงๆ แล้วมีทางแก้หลายทาง แต่มีทางหนึ่งที่ง่าย แม้จะไม่ใช่ทางที่ดีที่สุด คือการใส่ \n ไว้หน้า %c (ไม่ได้ใช้ได้ในทุกกรณี คอมไพล์เลอร์บางตัวจะไม่ยอม) ดังนั้นถ้าเจอปัญหานี้ค่อยแอบใช้

```c
#include<stdio.h>
void main() {
    char c1, c2, c3;
    scanf("%c", &c1);
    scanf("\n%c", &c2);
    scanf("\n%c", &c3);

    printf("Output: %c %c %c", c1, c2, c3);
}
```

```
A
B
C
Output: A B C
Process returned 13 (0xD)
Press any key to continue.
```

ข้อมูลชนิด character

ตัวอย่างโจทย์ (7)

จงเขียนโปรแกรมที่รับตัวอักขระจากผู้ใช้มาตัวหนึ่ง หากตัวอักขระนั้นเป็นตัวพิมพ์ใหญ่ ให้เปลี่ยนเป็นตัวพิมพ์เล็ก แล้วพิมพ์ผลลัพธ์ออกมาทางจอภาพ แต่หากไม่ใช่ตัวพิมพ์ใหญ่ ให้พิมพ์ตัวอักขระที่ได้มาจากผู้ใช้ออกมาทางจอภาพ

ข้อมูลชนิด character

ตัวอย่างโจทย์ (7)

```c
#include<stdio.h>
void main() {
    char c;
    scanf("%c", &c);
    if( c >= 65 && c <= 90 ) {
        c += 32;
    }
    printf("%c", c);
}
```

ข้อมูลชนิด character

ตัวอย่างโจทย์ (8)

โปรแกรมนี้ทำอะไร?

```c
#include<stdio.h>
void main() {
    char z;
    scanf("%c", &z);
    if( z >= 97 && z <= 122 ) {
        z -= 32;
    }
    printf("%c", z);
}
```

ข้อมูลชนิด character

ตัวอย่างโจทย์ (9)

หาข้อผิดพลาด

ข้อนี้ error หรือไม่ บรรทัดใด

```
1   #include<stdio.h>
2   void main() {
3       char z;
4       scanf("%c", 'z');
5       printf("%c", 'z');
6   }
```

ข้อมูลชนิด character

ตัวอย่างโจทย์ (10)

จงเขียนโปรแกรมรับตัวอักขระจากผู้ใช้มาตัวหนึ่ง หากตัวอักขระนั้นเป็นตัวพิมพ์ใหญ่ ให้เปลี่ยนเป็นตัวพิมพ์เล็ก แต่หากตัวอักขระจากผู้ใช้เป็นตัวพิมพ์เล็ก ให้เปลี่ยนเป็นตัวพิมพ์ใหญ่ แล้วพิมพ์ผลการเปลี่ยนออกมาทางจอภาพ หากตัวอักขระที่ได้มาไม่ใช่ตัวอักษรภาษาอังกฤษ (ไม่ใช่ตัวA-Z หรือ a-z) ให้พิมพ์ตัวอักขระที่ได้มาจากผู้ใช้ออกมามาทางจอภาพ

ข้อมูลชนิด character

ตัวอย่างโจทย์ (10)

```c
#include<stdio.h>
void main() {
    char c;
    scanf("%c", &c);
    if( c >= 65 && c <= 90 ) {
        c += 32;
    }
    else if( c >= 97 && c <= 122 ) {
        c -= 32;
    }
    printf("%c", c);
}
```

ข้อมูลชนิด character

ตัวอย่างโจทย์ (11)

จงเขียนโปรแกรมที่เปลี่ยนตัวอักขระเป็นตัวถัดไป โดยหากผู้ใช้พิมพ์ตัวอักขระมาเป็น A โปรแกรมจะเปลี่ยนเป็น B ถ้าผู้ใช้ใส่ตัวอักขระมาเป็น B, C, D, ..., Y โปรแกรมจะเปลี่ยนเป็น C, D, E, ..., Z แต่ถ้าผู้ใช้ใส่ตัว Z เข้ามาโปรแกรมจะเปลี่ยนเป็น A

ในกรณีที่ผู้ใช้ใส่ตัวพิมพ์เล็ก โปรแกรมจะเปลี่ยนตัวอักขระให้เป็นตัวถัดไปในลักษณะเดียวกัน (เปลี่ยน a เป็น b, b เป็น c, ..., z เป็น a) สุดท้ายโปรแกรมจะแสดงผลการเปลี่ยนตัวอักขระออกมาทางจอภาพ

ข้อมูลชนิด character

ตัวอย่างโจทย์ (11) แบบใช้รหัส ASCII

```
#include<stdio.h>
void main() {
    char c;
    scanf("%c", &c);
    if(c >= 65 && c <= 89 || c >= 97 && c <= 121) {
        c ++;
    }
    else if(c==90 || c==122){
        c -= 25;
    }
    printf("%c", c);
}
```

ไม่สนใจตัว ‘Z’ และ ‘z’

กรณีตัว ‘Z’ และ ‘z’ จะมาทำตรงนี้แทน

ข้อมูลชนิด character

ตัวอย่างโจทย์ (11) กรณีไม่ใช้รหัส แต่ใช้อักขระกันตรง ๆ เลย

```c
#include<stdio.h>
void main() {
    char c;
    scanf("%c", &c);
    if(c >= 'A' && c <= 'Y' || c >= 'a' && c <= 'y') {
        c ++;
    }
    else if(c=='Z' || c=='z'){
        c -= 25;
    }
    printf("%c", c);
}
```

Outline

- ฟังก์ชันไลบรารี่มาตรฐาน
  - นิยาม
  - Header File
  - ฟังก์ชันคณิตศาสตร์

- **สายอักขระ (สตริง; String)**
  - ข้อมูลชนิดอักขระ (character)
  - **ฟังก์ชันที่เกี่ยวข้องกับ character**
  - สตริง (String)
  - การจัดการ String

ฟังก์ชันไลบรารี่มาตรฐาน: ctype.h

สามารถเรียกใช้ได้หลัง include ไลบรารี่ ctype.h แล้ว

Function	แม่แบบ	ข้อมูลเบื้องต้น
islower	int **islower** (int c);	ทดสอบว่าเป็นพิมพ์**เล็ก**หรือไม่ (0 คือไม่, เลขอื่น คือจริง)
Isupper	int **isupper** (int c);	ทดสอบว่าเป็นพิมพ์**ใหญ่**หรือไม่ (0 คือไม่, เลขอื่น คือจริง)
tolower	int **tolower** (int c);	แปลงเป็นตัวพิมพ์**เล็ก** (คืนค่ารหัสอักขระของตัวพิมพ์**เล็ก**)
toupper	int **toupper** (int c);	แปลงเป็นตัวพิมพ์**ใหญ่** (คืนค่ารหัสอักขระของตัวพิมพ์**ใหญ่**)

ฟังก์ชันไลบรารี่มาตรฐาน: ctype.h

ตัวอย่างการใช้งานฟังก์ชันไลบรารี่มาตรฐาน ctype.h

```c
#include <stdio.h>
#include <ctype.h>
void main() {
    int c = 'A';
    printf("islower = %d\n", islower( c ) );
    printf("isupper = %d\n", isupper( c ) );
    printf("tolower = %c\n", tolower( c ) );
    printf("toupper = %c\n", toupper( c ) );
}
```

```
islower = 0
isupper = 1
tolower = a
toupper = A
```

ฟังก์ชันไลบรารี่มาตรฐาน: ctype.h

ทดสอบความเข้าใจ โจทย์: อ.ภิญโญ แท้ประสาทสิทธิ์

```c
#include <stdio.h>
#include <ctype.h>
void main() {
    int c = 'A';
    printf("islower = %d\n", islower( c ) );
    printf("isupper = %d\n", isupper( c ) );
    printf("tolower = %c\n", tolower( c ) );
    printf("toupper = %c\n", toupper( c ) );
}
```

- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘a’ จะได้ผลลัพธ์เป็นอย่างไร
- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘+’ จะได้ผลลัพธ์เป็นอย่างไร
- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘ ’ จะได้ผลลัพธ์เป็นอย่างไร
- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘\n’ จะได้ผลลัพธ์เป็นอย่างไร

ฟังก์ชันไลบรารี่มาตรฐาน: ctype.h

ทดสอบความเข้าใจ โจทย์: อ.ภิญโญ แท้ประสาทสิทธิ์

- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘a’ จะได้ผลลัพธ์เป็นอย่างไร
- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘+’ จะได้ผลลัพธ์เป็นอย่างไร
- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘ ’ จะได้ผลลัพธ์เป็นอย่างไร
- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘\n’ จะได้ผลลัพธ์เป็นอย่างไร

```
islower = 2
isupper = 0
tolower = a
toupper = A
```

```
islower = 0
isupper = 0
tolower = +
toupper = +
```

```
islower = 0
isupper = 0
tolower =
toupper =
```

```
islower = 0
isupper = 0
tolower =

toupper =
```

ฟังก์ชันไลบรารี่มาตรฐาน: ctype.h

ทดสอบความเข้าใจ โจทย์: อ.ภิญโญ แท้ประสาทสิทธิ์

```c
#include <stdio.h>
#include <ctype.h>
void main() {
    int c = 'A';
    printf("islower = %d\n", islower( c ) );
    printf("isupper = %d\n", isupper( c ) );
    printf("tolower = %c\n", tolower( c ) );
    printf("toupper = %c\n", toupper( c ) );
}
```

- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘a’ จะได้ผลลัพธ์เป็นอย่างไร
- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘+’ จะได้ผลลัพธ์เป็นอย่างไร
- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘ ’ จะได้ผลลัพธ์เป็นอย่างไร
- ถ้าเปลี่ยน c จาก ‘A’ ไปเป็น ‘\n’ จะได้ผลลัพธ์เป็นอย่างไร

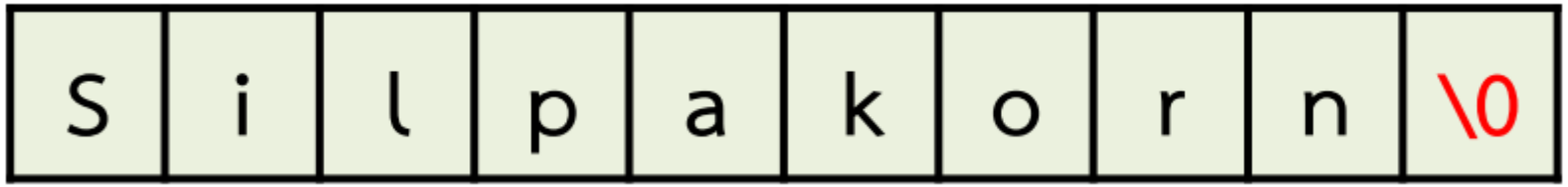
Outline

- ฟังก์ชันไลบรารี่มาตรฐาน
  - นิยาม
  - Header File
  - ฟังก์ชันคณิตศาสตร์

- **สายอักขระ (สตริง; String)**
  - ข้อมูลชนิดอักขระ (character)
  - ฟังก์ชันที่เกี่ยวข้องกับ character
  - **สตริง (String)**
  - การจัดการ String

ข้อมูลชนิด string

- สตริงมีชื่อเรียกที่นิยมอีกสองชื่อคือ ‘สายอักขระ’ และ ‘ข้อความ’
  - สตริงเป็นชนิดข้อมูลขั้นสูง เป็นการนำตัวอักขระมาต่อกันในอาเรย์ให้กลายเป็นข้อความ
- อาเรย์หนึ่งช่องจะมีตัวอักษรหนึ่งตัว
- อาเรย์ของสตริงจะมีอักขระศูนย์ (null character) มาปิดท้าย
  - **ดังนั้นต้องประกาศเผื่อไว้ 1 ช่อง**
- เช่น ถ้าเราต้องการเก็บคำว่า Silpakorn ข้างในอาเรย์ที่เป็นสตริงเป็นดังนี้



ข้อมูลชนิด string

การประกาศตัวแปรชนิด string

สามารถทำได้หลายวิธี

วิธีการที่ 1; นิยมที่สุด

```
char myString[256] = “Silpakorn”;
```

- สังเกตการใช้เครื่องหมาย “ ” แทนที่จะเป็น ‘ ’
- วิธีข้างบนจะสร้างพื้นที่เก็บอักขระไว้มากถึง 256 ตัว และปิดท้ายสตริงด้วยอักขระศูนย์ให้เราอัตโนมัติ
- จำนวน 256 ตัวที่ว่านับรวมอักขระศูนย์ด้วย ดังนั้นแท้จริงมันเก็บอักขระที่เราต้องการแสดงผลได้เพียง 255 ตัว
- วิธีข้างบนเป็นที่นิยมเพราะว่าพื้นที่เก็บข้อมูลมีความยาวพอที่จะเปลี่ยนค่าในอนาคตเป็นอย่างอื่นได้หลากหลาย โดยมีพื้นที่พอที่จะเก็บทั้งสตริงใหม่และอักขระศูนย์

ข้อมูลชนิด string

วิธีที่ 1

จากการประกาศ char myString[256] = “Silpakorn”;

เราจะได้อาเรย์ที่ภายในประกอบด้วยตัวอักขระดังนี้

อักขระในช่องอาเรย์	S	i	l	p	a	k	o	r	n	\0	?	...	?
ดัชนีในอาเรย์	0	1	2	3	4	5	6	7	8	9	10	...	255

- จะเห็นได้ว่ามีอักขระศูนย์มาต่อท้ายที่ตำแหน่งที่ 9 ให้อัตโนมัติ
- มีที่ว่างที่ไม่ได้ใช้อยู่เยอะมาก (แทนด้วย ? ในภาพด้านบน) แต่มันก็ทำให้เราสามารถเปลี่ยนข้อความในสตริงได้โดยง่าย
- การเตรียมที่ว่างไว้โดยปรกติเป็นเรื่องดี เพราะเราไม่สามารถเปลี่ยนความยาวอาเรย์ได้ในภาษา C แต่บางทีเราอยากเปลี่ยนตัวอักขระในอาเรย์ เช่น วันดีคืนดี อยากเติมคำว่า university เป็นต้น

ข้อมูลชนิด string

วิธีที่ 2 ไม่กำหนดขนาด

บางทีเราก็ไม่ต้องการเปลี่ยนค่าสตริง ดังนั้นก็ไม่จำเป็นที่จะต้องเผื่อพื้นที่เอาไว้

ในกรณีนี้ให้เราประกาศสตริงเป็นอาเรย์ของอักขระโดยที่ไม่ต้องกำหนดขนาด ดังนี้

```
char myString [ ] = “Silpakorn”;
```

- จากตัวอย่างข้างบนเราจะได้อาเรย์ของอักขระที่กำหนดพร้อมกับอักขระศูนย์ปิดท้ายให้อัตโนมัติในลักษณะเดิม
- อาเรย์ที่ได้จึงมีความยาวทั้งหมดสิบอักขระ (เพราะรวมอักขระศูนย์ด้วย)

ข้อมูลชนิด string

วิธีที่ 2 ไม่กำหนดขนาด

จากการประกาศด้วยวิธีที่ 2 หรือ char myString [ ] = “Silpakorn”;

- จะได้อาเรย์ของ char ดังนี้

อักขระในช่องอาเรย์	S	i	l	p	a	k	o	r	n	\0
ดัชนีในอาเรย์	0	1	2	3	4	5	6	7	8	9

- สังเกตว่าขนาดของอาเรย์พอดีกับข้อความ ดังนั้นถ้าจะเปลี่ยนข้อความ จะทำได้เพียงเปลี่ยนให้สั้นลงหรือมีความยาวเท่าเดิมเท่านั้น (เติมคำว่า university ไม่ได้แล้ว) ขยายขนาดทีหลังก็ทำไม่ได้

ข้อมูลชนิด string

วิธีที่ 3 ไม่กำหนดขนาด แต่ใช้ pointer

เนื่องจากอาเรย์เป็นพอยเตอร์ เราจึงสามารถประกาศสตริงโดยใช้พอยเตอร์ได้

```
char *myString = “Silpakorn”;
```

- เหมือนวิธีที่ 2 ทุกประการ

อักขระในช่องอาเรย์	S	i	l	p	a	k	o	r	n	\0
ดัชนีในอาเรย์	0	1	2	3	4	5	6	7	8	9

ข้อมูลชนิด string

วิธีอื่นๆ ที่ทำได้แต่ไม่นิยม

char myString[ ] = {‘S’, ‘i’, ‘l’, ‘p’, ‘a’, ‘k’, ‘o’, ‘r’, ‘n’};

*สังเกตว่าเมื่อเป็น char รายตัวอักษร จะใช้ ‘ ’ แต่ถ้าเป็น string ทั้งก้อน จะใช้ “ ”

อักขระในช่องอาเรย์	S	i	l	p	a	k	o	r	n
ดัชนีในอาเรย์	0	1	2	3	4	5	6	7	8

*สังเกตว่า ไม่มีอักขระ 0 ปิดท้าย ทำให้ไม่เหมาะกับการนำไปใช้ในรูปแบบ string

char myString[ ] = {‘S’, ‘i’, ‘l’, ‘p’, ‘a’, ‘k’, ‘o’, ‘r’, ‘n’, ‘\0’};

ถ้าใส่เองแบบนี้ก็ทำได้ แต่ผลที่ได้ก็เหมือนกับวิธี 1, 2, 3

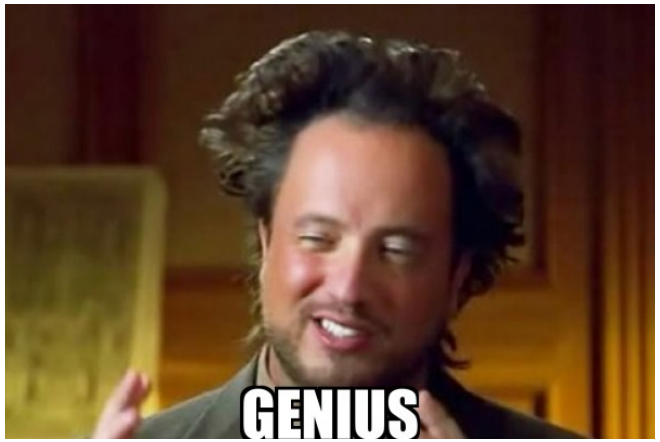
ข้อมูลชนิด string

วิธีอื่นๆ ที่ทำได้แต่ไม่นิยม

```
char myString[10] = {“Silpakorn”};  //ทำได้ แต่ปีกกาฟุ่มเฟือย
```

หรือจะใช้วิธีทรงพลัง

```
char myString [10];
myString[0] = ‘S’;      myString[1] = ‘i’;      myString[2] = ‘l’;
myString[3] = ‘p’;      myString[4] = ‘a’;      myString[5] = ‘k’;
myString[6] = ‘o’;      myString[7] = ‘r’;      myString[8] = ‘n’;
myString[9] = ‘\0’;     //ทำไปทำไม
```



ข้อมูลชนิด string

วิธีการประกาศที่ผิด: ทดสอบความเข้าใจ

1. char myString[9] = “Silpakorn”;
2. char myString[] = ‘Silpakorn’;
3. char myString[] = {“S”, “i”, “l”, “p”, “a”, “k”, “o”, “r”, “n”, “\0”};
4. char myString[] = Silpakorn;
5. char myString[] = {‘S’, ‘i’, ‘l’, ‘p’, ‘a’, ‘k’, ‘o’, ‘r’, ‘n’};

ข้อมูลชนิด string

วิธีการประกาศที่ผิด: ทดสอบความเข้าใจ

1. char myString[9] = “Silpakorn”; // ต้องเผื่อที่เก็บอักขระ 0 ด้วย
2. char myString[] = ‘Silpakorn’; // ใช้ “ ” แทน
3. char myString[] = {“S”, “i”, “l”, “p”, “a”, “k”, “o”, “r”, “n”, “\0”}; // ใช้ ‘ ’
4. char myString[] = Silpakorn; // ลืม “ ”
5. char myString[] = {‘S’, ‘i’, ‘l’, ‘p’, ‘a’, ‘k’, ‘o’, ‘r’, ‘n’}; // ไม่มีอักขระ 0 ปิดท้าย

ข้อมูลชนิด string

การรับและแสดงผล string

- เราใช้ scanf คู่กับ %s (ถ้าเป็นอักขระเราใช้ %c)
- แต่มีข้อสังเกตอยู่พอสมควร
  - สตริงเป็นอาเรย์ของอักขระ ดังนั้นเวลาใส่ค่าเข้าไป แท้จริงเราใส่ค่าเข้าไปในอาเรย์หลายช่องพร้อม ๆ กัน
- ตัวแปรอาเรย์อยู่ในฐานะตัวชี้ (pointer) เวลารับค่าจาก **scanf จะไม่ต้องใช้ &** เพราะเป็น address อยู่แล้ว
- ตอนแสดงผลเราใช้ printf คู่กับ %s
- ไม่ต้องกังวลว่ามันเป็นอาเรย์ ขอแค่มีอักขระศูนย์ปิดท้ายทุกอย่างจะดูดี
- เราใส่ชื่อตัวแปรสตริงลงในใน printf เหมือนตัวแปรทั่วไป

ข้อมูลชนิด string

การรับและแสดงผล string

```c
#include <stdio.h>
#include <ctype.h>
void main() {
    char firstName[256];
    char lastName[256];
    printf("Enter your first name: ");
    scanf("%s", firstName);
    printf("Enter your last name: ");
    scanf("%s", lastName);
    printf("Your name is %s %s", firstName,lastName);
}
```

ข้อมูลชนิด string

การดำเนินการทางตัวเลขกับ string

- เนื่องจากสตริงประกอบขึ้นมาจากอักขระ
- การดำเนินการทางตัวเลขที่ทำกับอักขระได้ก็จะทำบนสตริงได้ในลักษณะเดียวกัน
- แต่เราไม่สามารถทำกับทุกอักขระพร้อมๆกันได้ แต่สามารถดึงค่าอักขระออกมาประมวลผลทีละตัวได้
- การดึงอักขระออกมาทำได้เหมือนกับการอ่านค่าจากอาเรย์ทั่ว ๆ ไป เช่น
  - char c0 = str[0]; หมายถึงการดึงอักขระตัวแรกจาก str ไปเก็บไว้ที่ c0
  - str[3] = c0; หมายถึงการกำหนดให้อักขระตัวที่สี่ใน str มีค่าเท่ากับ c0
- หลังจากได้อักขระมาแล้ว เราสามารถดำเนินการบวกลบค่าตัวเลขกับอักขระหรือเปรียบเทียบค่าตัวเลขได้ในลักษณะเดียวกับที่เราทำกับอักขระทั่วไป จากนั้นใช้ loop ดำเนินการต่อเนื่องไป

ข้อมูลชนิด string

การดำเนินการทางตัวเลขกับ string : โจทย์ที่พบบ่อย

- การเปลี่ยนอักขระในสตริงให้เป็นตัวใหญ่ให้หมดทุกตัว
- การเปลี่ยนอักขระในสตริงให้เป็นตัวเล็กให้หมดทุกตัว
- การเปลี่ยนอักขระที่ต้นประโยคให้เป็นตัวใหญ่

ข้อมูลชนิด string

ตัวอย่างโจทย์ (12)

จงเขียนโปรแกรมที่ทำการเปลี่ยนตัวอักษรทุกตัวในข้อความให้กลายเป็นตัวพิมพ์ใหญ่ โดยที่

- ถ้าหากอักษรในข้อความตัวใดเป็นตัวเล็ก ให้โปรแกรมเปลี่ยนเป็นตัวใหญ่
- ถ้าตัวอักษรเป็นตัวพิมพ์ใหญ่อยู่แล้วให้คงไว้เช่นเดิม
- ถ้าเป็นตัวอักษรอื่น ๆ เช่น ตัวเลข สัญลักษณ์ เครื่องหมายวรรคตอนรวมทั้งอักขระพิเศษ ให้คงไว้เช่นเดิม
- กำหนดให้ข้อความที่ผู้ใช้ป้อนเข้ามามีตัวอักษรไม่เกิน 1023 ตัวอักษร

ข้อมูลเข้า	ผลลัพธ์
Silpakorn	SILPAKORN
CP#	CP#

ข้อมูลชนิด string

ตัวอย่างโจทย์ (12)

```c
#include <stdio.h>
#include <ctype.h>
void main() {
    char str[1024];
    scanf("%s", str);
    int i = 0;
    while( str[i] != '\0' ) {
        if(str[i] >= 'a' && str[i] <= 'z') {
            str[i] -= 32;
        }
        ++i;
    }
    printf("%s", str);
}
```

ทำซ้ำเรื่อยๆ เรียงไปตั้งแต่ตัวแรก ทีละตัวๆ จนกว่าจะพบอักขระ 0

ตรวจสอบว่าเป็นตัวพิมพ์เล็กหรือไม่ ถ้าไม่ใช่จะไม่เปลี่ยนค่า

ถ้าเป็นพิมพ์เล็ก จะลดรหัสลงไป 32 ทำให้ตัวแปร c เปลี่ยนรหัสไปเป็นตัวพิมพ์ใหญ่แทน

ข้อมูลชนิด string

ข้อจำกัดของ scanf กับ string

- เรารู้อยู่แล้วว่า scanf จะตัดข้อความเมื่อพบช่องว่าง (ไม่ว่าจะเป็น tab, enter หรือ space)
- นั่นทำให้เราสามารถนำเข้าข้อมูลตัวเลขทีละหลาย ๆ ตัวได้ เช่น พิมพ์ 2 4 5 6 9 แล้วจึง enter
  - ตัวเลขทั้ง 5 ตัวนี้จะเข้าไปอยู่ในตัวแปร 5 ตัว เรียงตามลำดับของ scanf
- ลองจินตนาการว่า ถ้าเราใช้ scanf รับค่า string โดยพิมพ์ว่า I love you จะให้ผลอย่างไร

ข้อมูลชนิด string

ข้อจำกัดของ scanf กับ string

```c
#include <stdio.h>
#include <ctype.h>
void main() {
    char str[1024];
    scanf("%s", str);
    printf("%s", str);
}
```

```
Silpakorn
Silpakorn
```

ถ้าไม่มีเว้นวรรคก็ไม่มีปัญหา

```
I love you
I
```

ถ้ามีเว้นวรรคจะรับแค่ตัวแรกเท่านั้น

```
Hello world
Hello
```

```
No problem?
No
```

ข้อมูลชนิด string

ข้อจำกัดของ scanf กับ string

- อาจแก้ไขปัญหานี้โดยการเปลี่ยนจากการใช้ scanf เป็น gets แทน
  - gets จะตัดคำเฉพาะเมื่อพบการกด enter หรือขึ้นบรรทัดใหม่เท่านั้น จะไม่ตัดเมื่อเจอเว้นวรรค
  - จริงๆแล้วมีหลายคำสั่งที่สามารถใช้ได้ แต่ gets เป็นตัวที่เข้าใจง่ายที่สุดแล้ว
- วิธีการใช้ gets นั้นง่ายกว่า scanf ด้วยซ้ำ แต่ใช้ได้เฉพาะข้อมูลประเภท string เท่านั้น
  - สามารถพิมพ์ gets (ชื่อตัวแปรประเภท string) ได้เลย ดังนี้

```
gets(c);  // ในกรณีตัวแปร string ชื่อ c
```

ข้อมูลชนิด string

ตัวอย่างการใช้ gets

```c
#include <stdio.h>
void main() {
    char str[1024];
    gets(str);
    printf("%s", str);
}
```

```
Silpakorn
Silpakorn
```

```
I love you
I love you
```

```
Hello world
Hello world
```

```
No problem?
No problem?
```

ข้อมูลชนิด string

ตัวอย่างโจทย์ (13)

จงเขียนโปรแกรมที่ทำการเปลี่ยนตัวอักษรทุกตัวในข้อความให้กลายเป็นตัวพิมพ์ใหญ่ โดยที่ข้อความได้มาจากผู้ใช้และอาจมีช่องว่างหรือตัวกั้นหน้ารวมอยู่ด้วย กำหนดให้ข้อความจากผู้ใช้สิ้นสุดเมื่อผู้ใช้สั่งขึ้นบรรทัดใหม่

ข้อมูลเข้า	ผลลัพธ์
Silpakorn	SILPAKORN
CP#	CP#
You have to fight.	YOU HAVE TO FIGHT.
That's incredible!	THAT'S INCREDIBLE!

ข้อมูลชนิด string

ตัวอย่างโจทย์ (13)

```c
#include <stdio.h>
void main() {
    char str[1024];
    gets(str);
    int i = 0;
    while(str[i] != '\0') {
        if(str[i] >= 97 && str[i] <= 122) {
            str[i] -= 32;
        }
        ++i;
    }
    printf("%s", str);
}
```

Outline

- ฟังก์ชันไลบรารี่มาตรฐาน
  - นิยาม
  - Header File
  - ฟังก์ชันคณิตศาสตร์

- **สายอักขระ (สตริง; String)**
  - ข้อมูลชนิดอักขระ (character)
  - ฟังก์ชันที่เกี่ยวข้องกับ character
  - สตริง (String)
  - **การจัดการ String**

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

สามารถเรียกใช้โดยการ include header file ที่มีชื่อว่า string.h (#include<string.h>)

- มีฟังก์ชันที่มีประโยชน์มากอยู่หลายตัว เช่น
  - การหาความยาวของข้อความ
  - การเปรียบเทียบข้อความ
  - การนำสตริงสองอันมาต่อกัน
- ในทางทฤษฎี เราสามารถเขียนฟังก์ชันเหล่านี้ด้วยตัวเองได้
  - แต่ในเมื่อมีคนทำมาไว้ให้เราแล้ว เราก็ไม่จำเป็นต้องเขียนใหม่ ยกเว้นสอบแล้วลืมว่าเรียกยังไง
  - โค้ดใน string.h มักจะทำงานเร็วกว่าที่เราเขียนเอง
  - โค้ดใน string.h มีโอกาสผิดพลาดต่ำมาก (เหลือแค่เรียกได้ถูกต้อง)

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

สามารถเรียกใช้ได้หลัง include ไลบรารี่ string.h แล้ว

Function	แม่แบบ	ข้อมูลเบื้องต้น
strlen	int **strlen** (char str[ ]);	นับจำนวนตัวอักษรใน string
strcat	char* **strcat** (char str1[ ], char str2[ ]);	ต่อสตริง 2 สาย
strcmp	int **strcmp** (char str1[ ], str2[ ]);	เปรียบเทียบสตริงตาม ASCII
strcmpi	int **strcmpi** (char str1[ ], str2[ ]);	เปรียบเทียบสตริงตามพจนานุกรม

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

ตัวอย่างการใช้งาน strlen()

แม่แบบ: int **strlen** (char str[ ]);

```c
#include <stdio.h>
#include <string.h>
void main() {
    char myString[256] = "Silpakorn";
    int length = strlen(myString);
    printf("Length = %d", length);
}
```

```
Length = 9
```

*สังเกตว่า ไม่มีการนับอักขระศูนย์

**อาจใช้ในการระบุจำนวนรอบของ loop ในการดำเนินการกับอักขระ

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

ตัวอย่างการใช้งาน strcat()

แม่แบบ: char* **strcat** (char str1[ ], char str2[ ]);

ใช้ pointer มารับ

ถ้าไม่ใช่ pointer จะ error

(แบบไหนไปลองดูเอง)

```c
#include <stdio.h>
#include <string.h>
void main() {
    char str[256] = "Silpakorn";
    char str2[256] = "University";
    char* outString = strcat(str, str2);
    printf("%s", outString);
}
```

```
SilpakornUniversity
```

*สังเกตว่า ไม่เติมเว้นวรรคให้ อยากได้ต้องเติมเอง

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmp () มีรายละเอียดยิบย่อยพอสมควร

แม่แบบ: int **strcmp** (char str1[ ], str2[ ]);

- เป็นคำสั่งเปรียบเทียบอักขระแต่ละตัวของสตริงจากซ้ายไปขวาทีละตัว
- การเปรียบเทียบจะทำจากซ้ายไปขวาทีละตัวอักษรจนกว่าจะพบความแตกต่างของตัวอักษรใน string1 และ string2 หรือสิ้นสุดข้อความ
- ความแตกต่างวัดกันที่รหัสแอสกี ดังนั้นตัวพิมพ์ใหญ่และเล็กจะให้ความแตกต่างในการเปรียบเทียบ (case sensitive)

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmp ()

- เมื่อพบความแตกต่างของอักขระในสตริงทั้งสอง สิ่งที่เกิดขึ้นตามมาจะเป็นดังนี้

ให้คำตอบเป็น ลบ	ให้คำตอบเป็น บวก	ให้คำตอบเป็น ศูนย์
ตรวจสอบเรียงจากตัวแรก แล้วเจอตัวที่ string1 มี ASCII น้อยกว่า string2	ตรวจสอบเรียงจากตัวแรก แล้วเจอตัวที่ string1 มี ASCII มากกว่า string2	ทั้งสองสตริง เหมือนกันทุกประการ
เจออักขระศูนย์ใน string1 ก่อน	เจออักขระศูนย์ใน string2 ก่อน	

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmp ( ) ตัวอย่างวัดความเข้าใจ ผลลัพธ์เป็นลบหรือบวก

strcmp(“Hello world”, “hello World”);

strcmp(“hello world”, “hello World”);

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmp ( ) ตัวอย่างวัดความเข้าใจ ผลลัพธ์เป็นลบหรือบวก

strcmp(“Hello World”, “Hello World”);

strcmp(“Hello”, “Hello World”);

strcmp(“BOOT”, “BUT”);

strcmp(“Boot”, “BUt”);

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmp ( ) ตัวอย่างวัดความเข้าใจ ผลลัพธ์เป็นลบหรือบวก

strcmp(“Hello world”, “hello World”);

- จะได้ค่าเป็น -1 เพราะว่า H ของ string1 เป็นตัวพิมพ์ใหญ่ ในขณะที่ h ใน string2 เป็นตัวพิมพ์เล็ก
- เนื่องจากตัวพิมพ์ใหญ่มีค่าแอสกีน้อยกว่าตัวพิมพ์เล็ก จึงสรุปได้ว่า string1 < string2 และได้ค่าเป็นลบ

strcmp(“hello world”, “hello World”);

- จะได้ค่าเป็นบวก เพราะการเปรียบเทียบจะดำเนินไปทีละตัวจนกว่าจะพบความแตกต่าง ซึ่งอยู่ที่ตัวอักษร W
- อักขระใน string1 เป็นตัวพิมพ์เล็กจึงมีค่าแอสกีมากกว่า ผลลัพธ์จึงเป็นบวก

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmp ( ) ตัวอย่างวัดความเข้าใจ ผลลัพธ์เป็นลบหรือบวก

strcmp(“Hello World”, “Hello World”);

- ได้ค่าเป็นศูนย์ เพราะสตริงเหมือนกันทุกประการ

strcmp(“Hello”, “Hello World”);

- ได้ค่าเป็นลบ เพราะพบอักขระศูนย์ที่ string1 ก่อน string2

strcmp(“BOOT”, “BUT”);

- ได้ค่าเป็นลบ เพราะพบความแตกต่างที่ ‘O’ และ ‘U’ โดย ‘O’ มีค่ารหัส ASCII น้อยกว่า ‘U’

strcmp(“Boot”, “BUt”);

- ได้ค่าเป็นบวก เพราะพบความแตกต่างที่ ‘o’ และ ‘U’ โดย ‘o’ มีค่ารหัส ASCII มากกว่า ‘U’

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmpi () มีความคล้าย strcmp () แต่ตามพจนานุกรม ไม่ใช่ ASCII

แม่แบบ: int **strcmpi** (char str1[ ], str2[ ]);

- ตัวอักษรใน string1 และ string2 จะถูกมองว่ามีรหัสแอสกีเหมือนตัวพิมพ์เล็ก ทำให้การเปรียบเทียบตัวอักษร**ไม่มีความต่างกันระหว่างตัวพิมพ์ใหญ่และเล็ก**
- ใช้รหัสแอสกีของตัวพิมพ์เล็กในการเปรียบเทียบตัวอักษรอังกฤษกับเครื่องหมายวรรคตอนหรือสัญลักษณ์พิเศษ

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmpi () มีความคล้าย strcmp () แต่ตามพจนานุกรม ไม่ใช่ ASCII

แม่แบบ: int **strcmpi** (char str1[ ], str2[ ]);

- ตัวอักษรใน string1 และ string2 จะถูกมองว่ามีรหัสแอสกีเหมือนตัวพิมพ์เล็ก ทำให้การเปรียบเทียบตัวอักษร**ไม่มีความต่างกันระหว่างตัวพิมพ์ใหญ่และเล็ก**
- ใช้รหัสแอสกีของตัวพิมพ์เล็กในการเปรียบเทียบตัวอักษรอังกฤษกับเครื่องหมายวรรคตอนหรือสัญลักษณ์พิเศษ

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmpi () ตัวอย่างวัดความเข้าใจ

strcmpi(“Hello world”, “hello World”);

strcmpi(“hello world”, “hello World”);

strcmpi(“Hello World”, “Hello World”);

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmpi () ตัวอย่างวัดความเข้าใจ

strcmpi(“Boot”, “BUt”);

strcmpi(“bu^”, “BUT”);

strcmpi(“bu{“, “BUT”);

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmpi () ตัวอย่างวัดความเข้าใจ

strcmpi(“Hello world”, “hello World”);

ได้ค่า 0

strcmpi(“hello world”, “hello World”);

ได้ค่า 0

strcmpi(“Hello World”, “Hello World”);

ได้ค่า 0 เพราะไม่สนใจตัวพิมพ์ใหญ่หรือเล็ก

ฟังก์ชันไลบรารี่มาตรฐานสำหรับจัดการ string

การใช้งาน strcmpi () ตัวอย่างวัดความเข้าใจ

strcmpi(“Boot”, “BUt”);

ได้ค่า - เพราะ ‘o’ มีรหัส ASCII น้อยกว่า ‘u’ (มองเป็นตัวพิมพ์เล็กทั้งหมด)

strcmpi(“bu^”, “BUT”);

ได้ค่า – เพราะ ‘^’ มีรหัส ASCII น้อยกว่า ‘u’ (ต้องเปิดตารางหน่อย)

strcmpi(“bu{“, “BUT”);

ได้ค่า + เพราะ ‘{’ มีรหัส ASCII มากกว่า ‘t’ (ต้องเปิดตารางหน่อย)

สรุป

- ฟังก์ชันไลบรารี่มาตรฐาน
  - นิยาม
  - Header File
  - ฟังก์ชันคณิตศาสตร์

- **สายอักขระ (สตริง; String)**
  - ข้อมูลชนิดอักขระ (character)
  - ฟังก์ชันที่เกี่ยวข้องกับ character
  - สตริง (String)
  - **การจัดการ String**