การเขียนโปรแกรมคอมพิวเตอร์ 1

Computer Programming I

ฟังก์ชันไลบรารีมาตรฐาน (Standard Library Functions)

สตริง (String)

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

ภาควิชาคอมพิวเตอร์ คณะวิทยาศาสตร์ มหาวิทยาลัยศิลปากร

สัปดาห์ที่ 12

หัวข้อเนื้อหา

- ฟังก์ชันไลบรารีมาตรฐานคืออะไร
- Header File กับฟังก์ชันไลบรารีมาตรฐาน
- ฟังก์ชันคณิตศาสตร์
- ข้อมูลชนิดตัวอักขระ (character)
- ฟังก์ชันเกี่ยวกับตัวอักขระ
- สตริง
- ฟังก์ชันจัดการสตริง

รู้จักกับไลบรารีมาตรฐาน

- ไลบรารี (Library) คือศูนย์รวมชุดคำสั่ง (Function)
- เป็นเหมือนห้องสมุดที่รวมสิ่งที่มีคนเขียนไว้ก่อนหน้าให้เราอ้างอิงได้ทันที
- ไลบรารีมาตรฐาน คือ ศูนย์รวมชุดคำสั่งมาตรฐานที่มีในคอมไพเลอร์ภาษาซีเกือบทุกตัว
  - คอมไพเลอร์ภาษาซีที่ได้มาตรฐานจะมีไลบรารีเหล่านี้
  - ทำให้เราสามารถเปลี่ยนคอมไพเลอร์เป็นของยี่ห้ออื่นได้ (แต่ก็ต้องเป็นภาษาซีเหมือนเดิม)
- แท้จริงภาษาซีเป็นมากกว่าภาษา เพราะมันเป็นสิ่งที่ถูกกำหนดไว้เป็นมาตรฐานสากลด้วย (เป็นมาตรฐาน ISO แบบหนึ่ง)

ฟังก์ชันไลบรารีมาตรฐาน

- คือฟังก์ชันที่ถูกกำหนดไว้ในมาตรฐานภาษาซี
  - เรียกใช้ได้จากไลบรารีมาตรฐาน
  - เรียกใช้ได้จากคอมไพเลอร์หลายยี่ห้อ
- ตัวอย่างที่เราใช้มาตลอดก็คือ scanf กับ printf
  - ยังมีฟังก์ชันพื้นฐานที่สำคัญอีกมากที่เราจะได้ใช้บ่อย ๆ เช่น sqrt เป็นฟังก์ชันสำหรับหาค่ารากที่สองของตัวแปรหรือตัวเลข
  - ฟังก์ชันไลบรารีมาตรฐานมักจะเกี่ยวข้องกับงานพื้นฐานที่ขาดไม่ได้

ความสำคัญของไลบรารีมาตฐาน

- การใช้ไลบรารีมาตรฐานในโปรแกรมของเราจะทำให้การนำโปรแกรมไปใช้ในระบบปฏิบัติการอื่นเป็นไปโดยง่าย
  - เขียนโปรแกรมบนวินโดวส์ แต่สามารถนำไปคอมไพล์ใหม่ให้ทำงานบน Mac หรือ Linux ได้
  - การนำไปคอมไพล์ใหม่ในคอมไพลเลอร์ยี่ห้ออื่นได้ เรียกว่า “มี portability”
- ลดเวลาในการเขียนโปรแกรมใหม่ เพราะโปรแกรมที่เขียนขึ้นมาสามารถนำไปใช้ได้อย่างแพร่หลาย
- แต่ไลบรารีมาตรฐานก็ไม่ได้ครอบคลุมการทำงานทุกอย่าง
  - ในทางปฏิบัติเราต้องพึ่งไลบรารีที่ไม่ใช่มาตรฐานสากลด้วย
  - ไลบรารีที่ไม่ใช่มาตรฐานบางอันก็สามารถนำไปใช้ได้กว้างขวาง บางอันก็ไม่

Header File กับฟังก์ชันไลบรารีมาตรฐาน

- เวลาเราเขียนว่า #include <stdio.h> แท้จริงแล้วเรากำลังอ้างถึงไลบรารีมาตรฐานผ่านสิ่งที่เรียกว่า header file
- header file เป็นไฟล์ที่บรรจุการประกาศฟังก์ชันหรือค่าคงที่เอาไว้
  - โดยมากจะลงท้ายด้วย .h (แต่ก็ไม่แน่เสมอไป)
  - เมื่อเราสั่ง #include ไฟล์แบบนี้แล้วโปรแกรมเราจะสามารถเรียกฟังก์ชันหรือค่าคงที่ที่มีการประกาศเอาไว้
    → ทำให้โปรแกรมของเรามีความสามารถมากขึ้น
  - เช่นแต่ก่อนโปรแกรมเรารับข้อมูลเข้าและแสดงผลไม่ได้ แต่พอรวม stdio.h มาก็ทำให้โปรแกรมเรามีความสามารถเพิ่มขึ้น
- ไลบรารีมาตรฐานถูกแบ่งเป็นหมวดหมู่ตามเฮดเดอร์ไฟล์

เฮดเดอร์ไฟล์ที่สำคัญ

- ด้านคณิตศาสตร์ math.h
- ค่าคงที่สำคัญ (เช่น INT_MAX) limits.h
- อรรถประโยชน์ (เช่น ขอหน่วยความจำ เปลี่ยนตัวอักษรให้เป็นตัวเลข) stdlib.h
- ตัวอักขระ (character) ctype.h
- สายอักขระ (string) string.h

ฟังก์ชันคณิตศาสตร์ (1)

- เราสามารถเรียกฟังก์ชันคณิตศาสตร์ขั้นสูงได้จาก math.h
- **หมวดเลขยกกำลัง**
  - ฟังก์ชัน sqrt ใช้หารากที่สอง
    แม่แบบฟังก์ชัน double **sqrt** (double input);
  - ฟังก์ชัน pow ใช้คำนวณผลการยกกำลัง
    แม่แบบฟังก์ชัน double **pow** (double base, double exp);
  - ฟังก์ชัน exp ใช้คำนวณผลการยกกำลังโดยมีฐานเป็น e (~ 2.71828)
    แม่แบบฟังก์ชัน double **exp** (double arg);
  - ฟังก์ชัน log ใช้คำนวณค่า logarithm โดยมีฐานเป็น e
    แม่แบบฟังก์ชัน double **log** (double arg);

หัดใช้ฟังก์ชัน sqrt

- การที่เราจะใช้ฟังก์ชันพวกนี้ได้เราจะต้องรู้จักวิธีอ่านแม่แบบฟังก์ชัน double **sqrt** (double input);
- อย่าลืมว่าฟังก์ชันขึ้นต้นด้วย return type ซึ่งระบุชนิดข้อมูลที่ฟังก์ชันจะส่งกลับมา ในที่นี้คือเลขทศนิยมความแม่นยำสูง
- ส่วนตัวหลักก็คือชื่อฟังก์ชัน ซึ่งในที่นี้คือ sqrt
- อินพุตของฟังก์ชันเป็นข้อมูลชนิดเลขทศนิยมความแม่นยำสูง
- ดังนั้นเราสามารถหารากที่สองของ x และเก็บผลลัพธ์ไว้ใน result ได้โดย

  double result;

  result = sqrt(x);

  หรือจะเขียนรวบเป็นคำสั่งเดียวกันได้ผลเป็น double result = sqrt(x);

การประยุกต์ใช้ฟังก์ชัน sqrt

- หาความยาวด้านสามเหลี่ยมตรงข้ามมุมฉาก

$$c = \sqrt{a^2 + b^2}$$

- สูตรข้างบนแท้จริงแล้วมาจาก $c^2 = a^2 + b^2$

ตัวอย่าง จงเขียนโปรแกรมที่รับความยาวด้านประกอบมุมฉากทั้งสองเป็นจำนวนจริงบวก แล้วคำนวณหาความยาวด้านตรงข้ามมุมฉาก พร้อมแสดงผลการคำนวณออกมาทางจอภาพ

วิเคราะห์ จำนวนจริงหมายความรวมถึงทั้งจำนวนเต็มและเลขทศนิยม ซึ่งเทียบเคียงได้กับ float หรือ double

การให้ค่า a และ b เป็นเลขแบบ double นับเป็นทางที่ปลอดภัย เพราะเก็บข้อมูลได้มากกว่าและละเอียดกว่า

โค้ดการหาความยาวด้านสามเหลี่ยมตรงข้ามมุมฉาก

```
#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);
}
```

เราเรียกใช้ header file ได้หลายอันพร้อมกัน ในที่นี้โปรแกรมเราจะมีความสามารถด้านคณิตศาสตร์ขั้นสูง พร้อมทั้งรับข้อมูลและแสดงผลลัพธ์ได้

ถ้าเป็น double ให้ใช้ L เล็กหน้า f

เรียก sqrt เหมือนฟังก์ชันทั่วไป ค่าของพารามิเตอร์สามารถนำมาคำนวณที่ในวงเล็บก็ได้ (เหมือน printf)

การประยุกต์ใช้ฟังก์ชัน pow

- มีความจำเป็นเมื่อเลขชี้กำลังไม่เป็นจำนวนเต็ม เช่น ถ้าเราอยากรู้ว่า $3.14^{1.25}$มีค่าเท่าใด
- ใช้ได้ในกรณีที่เลขชี้กำลังมีค่ามาก หรือเลขชี้กำลังติดค่าตัวแปร เช่น ถ้าเราต้องการหาค่าของ 3.14^{y}หรือ x^{y}เป็นต้น
- ใช้ในการหาค่ารากที่สามของระบบสมการ

ตัวอย่าง จงหาค่า xเมื่อกำหนดให้ $y = x^3$โดยที่ yเป็นอินพุตจากผู้ใช้

วิเคราะห์ เราสามารถหาค่า x ได้จาก $x = \sqrt[3]{y} = y^{\frac{1}{3}}$

โค้ดแสดงการใช้ pow หาค่ารากที่สาม

- แม่แบบฟังก์ชัน double **pow** (double base, double exp);

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

void main() {
    double x, y;
    scanf("%lf", &y);

    x = pow(y, 1.0/3.0);
    printf("%lf", x);
}
```

เพื่อความง่ายในการพิจารณาความถูกต้อง ลองแทน y ให้เป็น 8 หรือ 125

ฟังก์ชันคณิตศาสตร์ (2)

หมวดตรีโกณมิติ

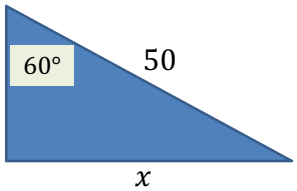
- ฟังก์ชันพื้นฐาน sin, cos, และ tan
  แม่แบบฟังก์ชัน double **sin** (double arg);
  cos กับ tan มีรูปแบบการใช้งานแบบเดียวกัน
- ฟังก์ชันหามุม asin, acos, atan, และ atan2 (คือการหาค่า arcsin, arccos, arctan แบบ quadrant เดียวและ arctan แบบ 2 quadrant)
  แม่แบบฟังก์ชัน double **asin** (double arg);
  asin, acos และ atan มีแม่แบบคล้ายกัน ต่างกันเฉพาะ atan2 เท่านั้น
  double **atan2** (double y, double x);

หน่วยของมุมเป็น**เรเดียน** ไม่ใช่องศา

atan2 ใช้เครื่องหมายของพารามิเตอร์ในการหา quadrant

การประยุกต์ใช้ฟังก์ชันตรีโกณมิติ

- หาความด้านสามเหลี่ยมที่ติดมุมฉาก



จงคำนวณหาความยาวของ x
เมื่อกำหนดความยาวด้านตรงข้ามมุมฉาก
และขนาดมุมมาให้ดังภาพ

- เรารู้ว่าเราต้องคำนวณ $50 * \sin(60°)$
- เรื่องที่เราต้องระวังในการใช้ฟังก์ชันตรีโกณมิติในภาษาซีก็คือว่า หน่วยของมุมไม่ใช่องศา แต่เป็นเรเดียน
- ต้องเปลี่ยนหน่วยให้เป็นเรเดียนด้วย $\frac{A}{180}\ \pi$ เมื่อ A เป็นมุมในหน่วยองศา

โค้ดตัวอย่างการใช้ฟังก์ชันตรีโกณมิติ

- เราสามารถอ้างถึงค่า π ได้จากค่าคงที่ชื่อว่า M_PI (มากับ math.h)

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

void main() {
    double angle = (60.0 / 180.0) * M_PI;
    printf("%lf", 50 * sin(angle));
}
```

- ถ้าจำชื่อค่าคงที่ไม่ได้ ก็ใช้ความรู้ทางคณิตศาสตร์ช่วยได้ เช่น เพราะว่า arctan(1) = $\frac{\pi}{4}$ ถ้าเราเขียนโค้ดว่า

  double pi = 4.0 * atan(1.0) เราก็จะได้ค่า π มาเหมือนกัน

การใช้กฎของ cosine

- เวลาที่เราคูณเวคเตอร์สองตัวด้วย dot product ตามสูตร

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

- สถานการณ์หนึ่งหนึ่งที่เราพบบ่อยก็คือ เรารู้เวคเตอร์ $\vec{u}$ และ $\vec{v}$ แต่เราอยากจะรู้ว่าเวคเตอร์ทั้งสองทำมุมกันเท่าไหร่ ดังนั้นเราจึงต้องทำการปรับสูตร ย้ายข้างกันสักเล็กน้อยเป็น

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

จากนั้นใช้ arccosเราก็จะได้ว่ามุม θ คือ

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

ตัวอย่างการใช้ acos หาค่ามุมของเวคเตอร์

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

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

1. เราสามารถคำนวณค่า dot product ได้จาก (3 x 1) + (2 x 4)
2. ขนาดของเวคเตอร์ทั้งสองหาได้จาก$\|\vec{u}\| = \sqrt{3^2 + 2^2}$และ $\|\vec{v}\| = \sqrt{1^2 + 4^2}$
3. ถ้าหาค่าของ $\frac{\vec{u} \cdot \vec{v}}{\|\vec{u}\|\|\vec{v}\|}$ ได้สำเร็จ เราก็นำใช้กับฟังก์ชัน acosเพื่อหาผลลัพธ์สุดท้าย (ได้ผลลัพธ์เป็นมุมในหน่วยเรเดียนเช่นเดิม)

โค้ดการหามุมระหว่างเวคเตอร์

```
#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);
}
```


ฟังก์ชันคณิตศาสตร์ (3)

หมวดเศษทศนิยม

- ฟังก์ชันปัดค่าขึ้น ceil

  แม่แบบฟังก์ชัน double **ceil** (double x);

  เช่น ถ้า x = 1.23 ผลที่ได้คือ 2.00

  ถ้า x = -1.23 ผลที่ได้คือ -1.00 (**เพราะว่า -1.00 มีค่ามากกว่า -2.00**)

- ฟังก์ชันปัดค่าลง floor

  แม่แบบฟังก์ชัน double **floor** (double x);

  เช่น ถ้า x = 1.23 ผลที่ได้คือ 1.00

  ถ้า x = -1.23 ผลที่ได้คือ -2.00 (**เพราะว่า -2.00 มีค่าน้อยกว่า -1.00**)

ตัวอย่างการใช้ ceil และ floor

```
#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);
}
```


ข้อมูลชนิดตัวอักขระ

- ที่ผ่านมาเราเน้นการทำงานกับข้อมูลชนิดจำนวนเต็มกับจำนวนจริง
- แต่ข้อมูลชนิดตัวอักขระ (character) ก็มีความสำคัญมาก
  - เป็นพื้นฐานของการเก็บข้อมูลจำพวกชื่อและข้อความต่าง ๆ
  - เป็นพื้นฐานของการแสดงผลลัพธ์หลาย ๆ อย่าง
- คำว่าตัวอักขระในที่นี้หมายถึงตัวอักษรโดด
  - ไม่ใช่ข้อความ แต่เป็นเป็นตัวอักษรแค่ตัวเดียว
  - หมายความรวมถึงเครื่องหมายวรรคตอน และ ตัวเลข
  - รวมถึงตัวอักษรพิเศษที่มองไม่เห็น เช่น ช่องว่าง (space), ตัวขึ้นบรรทัดใหม่ (new line), หรือแม้กระทั่งเสียงเตือน (bell)

การใช้ตัวอักขระ

- ชนิดข้อมูลที่เขียนในภาษาซีคือ char
- เวลาประกาศตัวแปรเราทำแบบปรกติคือ
  - ชนิดข้อมูล char ตามด้วยชื่อตัวแปร เช่น char c;
  - เราระบุค่าเริ่มต้นพร้อมการประกาศได้เช่นเดียวกับตัวแปรทั่วไป แต่จะมีรูปแบบของการระบุตัวอักษรที่เป็นเอกลักษณ์ ไม่เหมือนตัวเลข
- การระบุตัวอักษรที่ต้องการ
  - ถ้าเราจะระบุให้ตัวแปรมีค่าเท่ากับตัว a เราไม่สามารถเขียนว่า char c = a; **แบบนี้ผิด** เพราะการเขียน a แบบนี้เหมือนชื่อตัวแปรทั่วไป
  - เราต้องครอบด้วยเครื่องหมาย อัญประกาศเดี่ยว (‘เขาเดี่ยว’) เช่น char c = ‘a’;

เรื่องที่คนมักจะงง

เวลาเราระบุค่าตัวอักษร ถ้าเราเจาะจงตัวใดตัวหนึ่ง เราก็ระบุลงไปได้เลย พร้อมเครื่องหมายอัญประกาศเดี่ยว เช่น char c = ‘a’;

- แต่ถ้าเรามีตัวแปรประเภท char อยู่สองตัว เช่น char x = ‘a’; และ char c; เราสามารถระบุให้ตัวแปร c มีค่าเท่ากับตัวแปร x ด้วยการเขียนว่า c = x; แบบนี้ได้
- จากตัวอย่างข้างบน c จะมีค่าเท่ากับอักขระ ‘a’

เครื่องหมายอัญประกาศเดี่ยวใช้ระบุตัวอักขระ ถ้าไม่ใส่จะหมายถึงตัวแปร

ตัวตนที่แท้จริงของตัวอักขระ

- ตัวอักขระแท้จริงแล้วอยู่ในฐานะ ‘รหัสตัวเลข’ คือ เครื่องเก็บตัวเลขเอาไว้ แต่ตอนแสดงผลมันเปลี่ยนตัวเลขเป็นตัวอักษรให้เราอ่านออก
- เป็นการเชื่อมต่อกันระหว่างรูปแบบที่เครื่องเข้าใจกับรูปแบบที่มนุษย์เข้าใจ
  - เครื่องเข้าใจแต่ตัวเลข (เก็บข้อมูลทุกอย่างเป็นเลขฐานสองอยู่ภายใน)
  - มนุษย์ใช้การอ่านข้อความหรือตัวอักษร ไม่ได้ถนัดการอ่านตัวเลข
  - เราจึงให้เครื่องเก็บตัวเลขไว้ แต่ตอนแสดงผลให้มันแปลงตัวเลขเป็นตัวอักษรให้เราอ่าน
- เพื่อให้การเก็บรหัสตัวเลขของอักขระเป็นสิ่งที่เข้าใจตรงกันจึงได้มีการกำหนดมาตรฐานขึ้นมา หนึ่งในมาตรฐานที่ใช้กันอย่างแพร่หลายคือ ASCII

รู้จักรหัส ASCII

- ย่อมาจาก American Standard Code for Information Interchange
- เขียนเป็นไทยว่า ‘แอสกี’ (ออกเสียงคล้ายคำว่า แอส – คี่)
- อักขระแต่ละตัวจะมีตัวเลขประจำอยู่ เช่น
  A มีค่าตัวเลขเท่ากับ 65 และ a มีค่าตัวเลขเท่ากับ 97
- **ตัวเล็กกับตัวใหญ่ถือว่าเป็นคนละตัวกัน**
- มีอักขระพิเศษประกอบอยู่ใน ASCII Code เช่น
  - ‘\0’ มีค่าตัวเลขเท่ากับ 0 ใช้สำหรับแทนจุดสิ้นสุดของข้อความ (สตริง) เรานิยมเรียกอักขระตัวนี้ว่า null character (อักขระศูนย์)
  - ‘\t’ มีค่าตัวเลขเท่ากับ 9 ใช้สำหรับแทนจุดตั้งระยะกั้นหน้า (tab)

ตารางรหัส ASCII

Dec	Hx	Oct	Char	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr	Dec	Hx	Oct	Html	Chr
0	0	000	NUL (null)	32	20	040		Space	64	40	100	@	@	96	60	140	`	`
1	1	001	SOH (start of heading)	33	21	041	!	!	65	41	101	A	A	97	61	141	a	a
2	2	002	STX (start of text)	34	22	042	"	"	66	42	102	B	B	98	62	142	b	b
3	3	003	ETX (end of text)	35	23	043	#	#	67	43	103	C	C	99	63	143	c	c
4	4	004	EOT (end of transmission)	36	24	044	$	$	68	44	104	D	D	100	64	144	d	d
5	5	005	ENQ (enquiry)	37	25	045	%	%	69	45	105	E	E	101	65	145	e	e
6	6	006	ACK (acknowledge)	38	26	046	&	&	70	46	106	F	F	102	66	146	f	f
7	7	007	BEL (bell)	39	27	047	'	'	71	47	107	G	G	103	67	147	g	g
8	8	010	BS (backspace)	40	28	050	(	(	72	48	110	H	H	104	68	150	h	h
9	9	011	TAB (horizontal tab)	41	29	051	)	)	73	49	111	I	I	105	69	151	i	i
10	A	012	LF (NL line feed, new line)	42	2A	052	*	*	74	4A	112	J	J	106	6A	152	j	j
11	B	013	VT (vertical tab)	43	2B	053	+	+	75	4B	113	K	K	107	6B	153	k	k
12	C	014	FF (NP form feed, new page)	44	2C	054	,	,	76	4C	114	L	L	108	6C	154	l	l
13	D	015	CR (carriage return)	45	2D	055	-	-	77	4D	115	M	M	109	6D	155	m	m
14	E	016	SO (shift out)	46	2E	056	.	.	78	4E	116	N	N	110	6E	156	n	n
15	F	017	SI (shift in)	47	2F	057	/	/	79	4F	117	O	O	111	6F	157	o	o
16	10	020	DLE (data link escape)	48	30	060	0	0	80	50	120	P	P	112	70	160	p	p
17	11	021	DC1 (device control 1)	49	31	061	1	1	81	51	121	Q	Q	113	71	161	q	q
18	12	022	DC2 (device control 2)	50	32	062	2	2	82	52	122	R	R	114	72	162	r	r
19	13	023	DC3 (device control 3)	51	33	063	3	3	83	53	123	S	S	115	73	163	s	s
20	14	024	DC4 (device control 4)	52	34	064	4	4	84	54	124	T	T	116	74	164	t	t
21	15	025	NAK (negative acknowledge)	53	35	065	5	5	85	55	125	U	U	117	75	165	u	u
22	16	026	SYN (synchronous idle)	54	36	066	6	6	86	56	126	V	V	118	76	166	v	v
23	17	027	ETB (end of trans. block)	55	37	067	7	7	87	57	127	W	W	119	77	167	w	w
24	18	030	CAN (cancel)	56	38	070	8	8	88	58	130	X	X	120	78	170	x	x
25	19	031	EM (end of medium)	57	39	071	9	9	89	59	131	Y	Y	121	79	171	y	y
26	1A	032	SUB (substitute)	58	3A	072	:	:	90	5A	132	Z	Z	122	7A	172	z	z
27	1B	033	ESC (escape)	59	3B	073	;	;	91	5B	133	[	[	123	7B	173	{	{
28	1C	034	FS (file separator)	60	3C	074	<	<	92	5C	134	\	\	124	7C	174			|
29	1D	035	GS (group separator)	61	3D	075	=	=	93	5D	135	]	]	125	7D	175	}	}
30	1E	036	RS (record separator)	62	3E	076	>	>	94	5E	136	^	^	126	7E	176	~	~
31	1F	037	US (unit separator)	63	3F	077	?	?	95	5F	137	_	_	127	7F	177	&#127;	DEL

Source: www.LookupTables.com

การดำเนินการเลขคณิตกับตัวอักขระ

- เพราะตัวอักขระแท้จริงเป็นตัวเลข
  - เราสามารถดำเนินการคำนวณทางตัวเลขกับมันได้
  - บวกลบค่าอักขระได้
  - เปรียบเทียบความมากน้อยของตัวเลขต่าง ๆ ได้

ตัวอย่าง เปลี่ยนตัวอักษรใหญ่ให้กลายเป็นเล็ก

- เรารู้ว่าตัวอักษรใหญ่ในภาษาอังกฤษมีค่าตัวเลขน้อยกว่าตัวอักษรเล็กอยู่ 32
- เรารู้ด้วยว่าตัวอักษรใหญ่มีค่าตั้งแต่ 65 ไปจนถึง 90
  ดังนั้นถ้าเรามีอักขระ c เราสามารถรู้ได้ว่ามันเป็นตัวอักขระใหญ่หรือไม่โดย

```
if ( c >= 65 && c <= 90 ) { ... }
```

โค้ดสำหรับการแปลงตัวอักษรใหญ่ให้กลายเป็นตัวอักษรเล็ก

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

เราทำการเปรียบเทียบ char ได้เหมือนตัวเลขทั่วไป

จะบวกค่าของ char ก็ยังได้

- ลองเปลี่ยนค่าเริ่มต้นตัวแปร c เป็น 'B', 'C' หรือ 'Z'
- ลองเปลี่ยนค่าเริ่มต้นตัวแปร c เป็น 'a', 'b', 'c' หรือ 'z'
- ลองเปลี่ยนค่าเริ่มต้นตัวแปร c เป็น '+', '-', '*' หรือ '/'

การรับค่าอักขระจากผู้ใช้

เราใช้คำสั่ง scanf ได้ตามปรกติ แต่ให้ใช้ %c ในการระบุประเภทข้อมูล

ตัวอย่าง

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

เรื่องควรทราบ

- ถึงแม้ว่าในโปรแกรมเราจะใช้ ' ' ครอบอักขระไว้ **แต่เวลารับข้อมูลจากผู้ใช้อย่าใส่ ' ' เข้าไปเป็นอันขาด**
- บนวินโดวส์เวลาที่เรากด Enter มันจะมีอักขระขึ้นบรรทัดใหม่กับขึ้นต้นบรรทัดใหม่แยกออกจากกัน **ทำให้มีตัวอักษรเกินมาตัวหนึ่ง** (ถ้าโปรแกรมรับอินพุตมาตัวเดียว ปัญหาตัวอักษรเกินจะไม่มีผลใด ๆ)

โปรแกรมรับและแปลงค่าตัวพิมพ์ตัวใหญ่เป็นตัวเล็ก

ตัวอย่าง จงเขียนโปรแกรมที่รับตัวอักขระจากผู้ใช้มาตัวหนึ่ง หากตัวอักขระนั้นเป็นตัวพิมพ์ใหญ่ ให้เปลี่ยนเป็นตัวพิมพ์เล็ก แล้วพิมพ์ผลลัพธ์ออกมาทางจอภาพ แต่หากไม่ใช่ตัวพิมพ์ใหญ่ ให้พิมพ์ตัวอักขระที่ได้มาจากผู้ใช้ออกมาทางจอภาพ

```
char c;
scanf("%c", &c);
if( c >= 65 && c <= 90 ) {
    c += 32;
}
printf("%c", c);
```


แบบฝึกหัด (1)

1. จงเขียนโปรแกรมรับตัวอักขระจากผู้ใช้มาตัวหนึ่ง หากตัวอักขระนั้นเป็นตัวพิมพ์ใหญ่ ให้เปลี่ยนเป็นตัวพิมพ์เล็ก แต่หากไม่ใช่ตัวพิมพ์ใหญ่ ให้พิมพ์ตัวอักขระที่ได้มาจากผู้ใช้ออกมามาทางจอภาพ

2. จงเขียนโปรแกรมรับตัวอักขระจากผู้ใช้มาตัวหนึ่ง หากตัวอักขระนั้นเป็นตัวพิมพ์ใหญ่ ให้เปลี่ยนเป็นตัวพิมพ์เล็ก แต่หากตัวอักขระจากผู้ใช้เป็นตัวพิมพ์เล็ก ให้เปลี่ยนเป็นตัวพิมพ์ใหญ่ แล้วพิมพ์ผลการเปลี่ยนออกมาทางจอภาพ หากตัวอักขระที่ได้มาไม่ใช่ตัวอักษรภาษาอังกฤษ (ไม่ใช่ตัว A-Z หรือ a-z) ให้พิมพ์ตัวอักขระที่ได้มาจากผู้ใช้ออกมามาทางจอภาพ

แบบฝึกหัด (2)

3. 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

โค้ดโปรแกรมเปลี่ยนตัวอักขระไปตัวที่อยู่ติดกัน

```
char inputChar, outputChar;
scanf("%c", &inputChar);

// Perform conversion (except Z and z)
if(inputChar >= 65 && inputChar < 90) {
  outputChar = inputChar + 1;
} else if(inputChar == 90) {
  outputChar = 65;
} else if(inputChar >= 97 && inputChar < 122) {
  outputChar = inputChar + 1;
} else if(inputChar == 122) {
  outputChar = 97;
} else {
  outputChar = inputChar; // No conversion needed
}
printf("%c\n", outputChar);
```


ที่จริงเราไม่ต้องจำตัวเลข ใช้อักขระแทนตัวเลขได้เลย

```
char inputChar, outputChar;
scanf("%c", &inputChar);

// Perform conversion (except Z and z)
if(inputChar >= 'A' && inputChar < 'Z') {
  outputChar = inputChar + 1;
} else if(inputChar == 'Z') {
  outputChar = 'A';
} else if(inputChar >= 'a' && inputChar < 'z'){
  outputChar = inputChar + 1;
} else if(inputChar == 'z') {
  outputChar = 'a';
} else {
  outputChar = inputChar; // No conversion needed
}
printf("%c\n", outputChar);
```


หัวข้อเนื้อหา

- ฟังก์ชันไลบรารีมาตรฐานคืออะไร
- Header File กับฟังก์ชันไลบรารีมาตรฐาน
- ฟังก์ชันคณิตศาสตร์
- ข้อมูลชนิดตัวอักขระ (character)
- **ฟังก์ชันเกี่ยวกับตัวอักขระ**
- สตริง
- ฟังก์ชันจัดการสตริง

ฟังก์ชันเกี่ยวกับตัวอักขระ

- เรียกใช้ได้จากเฮดเดอร์ #include <ctype.h>
- มีอยู่สี่ฟังก์ชันที่สำคัญเป็นพิเศษคือ

int islower(int c);	ทดสอบว่า c เป็นตัวพิมพ์เล็กหรือไม่
int isupper(int c);	ทดสอบว่า c เป็นตัวพิมพ์ใหญ่หรือไม่
int tolower(int c);	คำนวณตัวพิมพ์เล็กของ c
int toupper(int c);	คำนวณตัวพิมพ์ใหญ่ของ c

อธิบายการใช้งานฟังก์ชันอักขระ (1)

- เพราะตัวอักขระแท้จริงเป็นเลขจำนวนเต็ม ฟังก์ชันจึงรับค่าเป็นเลขจำนวนเต็ม (แต่ที่จริงเราส่ง char ไปเป็นพารามิเตอร์)
- ฟังก์ชัน islower คืนค่า ‘จริง’ มาให้เมื่อพารามิเตอร์เป็นตัวพิมพ์เล็ก และคืนค่า ‘เท็จ’ มาให้เมื่อพารามิเตอร์ไม่ใช่ตัวพิมพ์เล็ก
  - ค่า ‘จริง’ คือเลขที่ไม่ใช่ศูนย์ (จะติดลบก็เรียกว่า ‘จริง’) แต่โดยปรกติคอมไพเลอร์จะเลือกคืนเลขหนึ่งมาให้
  - ค่า ‘เท็จ’ คือเลขศูนย์ (มีเลขศูนย์เพียงตัวเดียวเท่านั้นที่แทนคำว่าเท็จ)
- ฟังก์ชัน isupper คืนค่า ‘จริง’ มาให้เมื่อพารามิเตอร์เป็นตัวพิมพ์ใหญ่ และคืนค่า ‘เท็จ’ มาให้เมื่อพารามิเตอร์ไม่ใช่ตัวพิมพ์ใหญ่
- ถ้าเราใส่พวกเครื่องหมายวรรคตอนเข้าไป จะได้ผลลัพธ์เป็นเท็จ

อธิบายการใช้งานฟังก์ชันอักขระ (2)

- ฟังก์ชัน tolower จะคำนวณดูว่า c เป็นตัวพิมพ์ใหญ่หรือไม่ ถ้าใช่ก็จะคืนรหัสตัวเลขที่เป็นตัวพิมพ์เล็กของมันมาให้ ถ้าไม่ใช่ก็จะได้ค่า c กลับคืนมา
  - ค่า c จะไม่เปลี่ยนแปลงเป็นอันขาดไม่ว่ากรณีใด
  - ถ้าเราต้องการรู้ผลการแปลงเป็นตัวพิมพ์เล็ก เราต้องใช้ตัวแปรไปรับผลที่ฟังก์ชัน tolower คืนมาให้ เช่น

```
char c = ‘A’;
char result = tolower( c );
```

result จะมีค่าเป็นตัว ‘a’

- ฟังก์ชัน toupper จะคำนวณดูว่า c เป็นตัวพิมพ์เล็กหรือไม่ ถ้าใช่ก็จะคืนรหัสตัวเลขที่เป็นตัวพิมพ์ใหญ่ของมันมาให้ ถ้าไม่ใช่ก็จะได้ค่า c กลับคืนมา
- ถ้าเราส่งพวกเครื่องหมายวรรคตอนไปให้ค่าได้คืนมาจะเป็นค่า c ที่เราส่งไป

ตัวอย่างการใช้ฟังก์ชันอักขระ

```
#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 ) );
}
```

ผลลัพธ์

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

ทดสอบความเข้าใจ

จากหน้าที่แล้ว ถ้าเรา

- 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

ปัญหาตัวอักษรเกินมาเมื่อกด Enter

ตัวอักษรที่เกินมานี้จะถูกโอนไปใส่ตัวแปรถัดไป

- ทำให้เรื่องมันน่าสับสนมากหากโปรแกรมรับค่าอินพุตหลายตัวจากผู้ใช้
- ถ้าไม่จำเป็นจริง ๆ ควรเลี่ยงการรับข้อมูลเข้าจากผู้ใช้ในรูปแบบตัวอักษร
- ควรใช้ข้อมูลชนิดข้อความแทนตัวอักขระ แล้วจึงแยกตัวอักษรออกมาเพิ่มเติมในภายหลัง

เพื่อป้องกันความสับสนกับความแตกต่างนี้ เราสามารถใช้สตริงมาเก็บข้อมูลตัวอักขระทั่วไปก็ได้

- ตัวแปรชนิดอักขระยังมีความสำคัญในการคำนวณและแสดงผลตามปรกติ
- แต่บทบาทในการรับข้อมูลเข้าจากผู้ใช้จะถูกลดลงไป

ไม่ใช่ทุกภาษา คอมไพเลอร์ หรือทุกระบบปฏิบัติการจะเป็นแบบนี้

หัวข้อเนื้อหา

- ฟังก์ชันไลบรารีมาตรฐานคืออะไร
- Header File กับฟังก์ชันไลบรารีมาตรฐาน
- ฟังก์ชันคณิตศาสตร์
- ข้อมูลชนิดตัวอักขระ (character)
- ฟังก์ชันเกี่ยวกับตัวอักขระ
- **สตริง**
- ฟังก์ชันไลบรารีมาตรฐานสำหรับจัดการสตริง

ข้อมูลชนิดสตริง

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

S	i	l	p	a	k	o	r	n	\0

  - ดังนั้นอาเรย์ต้องมีจำนวนช่องมากกว่าจำนวนตัวอักขระที่เราคิดเก็บจริง ๆ อยู่หนึ่งช่อง (ต้องเผื่อที่ไว้เก็บอักขระศูนย์ด้วย)

การประกาศตัวแปรสตริง

ทำได้หลายแบบ แต่ที่นิยมที่สุดคือ

char myString[256] = “Silpakorn”;

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

ตัวอย่างการประกาศสตริง

จากการประกาศchar 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

การประกาศตัวแปรสตริงแบบไม่กำหนดขนาด (1)

บางทีเราก็ไม่ต้องการเปลี่ยนค่าสตริง ดังนั้นก็ไม่จำเป็นที่จะต้องเผื่อพื้นที่เอาไว้

- ในกรณีนี้ให้เราประกาศสตริงเป็นอาเรย์ของอักขระโดยที่ไม่ต้องกำหนดขนาด เช่น **char myString[ ] = “Silpakorn”;**
- จากตัวอย่างข้างบนเราจะได้อาเรย์ของอักขระที่กำหนดพร้อมกับอักขระศูนย์ปิดท้ายให้อัตโนมัติในลักษณะเดิม
- อาเรย์ที่ได้จึงมีความยาวทั้งหมดสิบอักขระ (เพราะรวมอักขระศูนย์ด้วย)

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

- เวลาที่เราประกาศสตริงแบบนี้เรามักจะไม่เปลี่ยนข้อความข้างใน แต่ก็สามารถเปลี่ยนได้โดยไม่เกิดความผิดพลาด ถ้าหากว่าความยาวของข้อความอันใหม่น้อยกว่าหรือเท่ากับข้อความแรก

การประกาศตัวแปรสตริงแบบไม่กำหนดขนาด (2)

- เนื่องจากอาเรย์เป็นพอยเตอร์ เราจึงสามารถประกาศสตริงโดยใช้พอยเตอร์ได้
- วิธีนี้จะให้ผลแบบเดียวกับการประกาศโดยใช้อาเรย์โดยไม่กำหนดขนาด

ตัวอย่าง

char *myString = “Silpakorn”;

วิธีนี้เราจะได้อาเรย์เก็บคำว่า Silpakorn พร้อมอักขระศูนย์ปิดท้ายอัตโนมัติ

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

ผลที่ได้เหมือนกับวิธีที่ผ่านมาทุกอย่าง คืออาเรย์จะมีขนาดเท่ากับจำนวนตัวอักษรในข้อความบวกกับอักขระศูนย์

รูปแบบการประกาศอาเรย์ของอักขระ

- การประกาศสตริงเราจะได้อาเรย์ของอักขระซึ่งปิดท้ายด้วยอักขระศูนย์
- การปิดท้ายด้วยอักขระศูนย์เป็นสิ่งที่สำคัญมาก เพราะฟังก์ชันไลบรารีมาตรฐานที่ใช้จัดการสตริงใช้อักขระศูนย์หาจุดสิ้นสุดข้อความ
- แต่หากเราจำเป็นที่จะต้องใช้อาเรย์ของ char ทั่ว ๆ ไปเราก็ทำได้ เช่น

char myString[ ] = {‘S’, ‘i’, ‘l’, ‘p’, ‘a’, ‘k’, ‘o’, ‘r’, ‘n’};
(สังเกตด้วยว่าพอเป็นอักขระแล้วเราใช้ ‘ ’ แต่พอเป็นสตริงเราใช้ “ ”)

- วิธีประกาศแบบข้างบนนี้จะทำให้ได้อาเรย์ของ char ดังแสดงทางด้านใต้

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

- สังเกตด้วยว่าไม่มีอักขระศูนย์ปิดท้าย ทำให้ไม่เหมาะกับการใช้งานเป็นข้อความ

การประกาศตัวแปรสตริงแบบอื่น ๆ

ยังมีวิธีประกาศหรือสร้างสตริงที่มีอักขระศูนย์ปิดท้ายแบบอื่น ๆ อยู่ แต่วิธีพวกนี้เต็มไปด้วยความรุ่มร่ามไม่กะทัดรัดจึงไม่เป็นที่นิยม

- char myString[10] = {“Silpakorn”};
  ที่จริงวงเล็บปีกกาที่จริงไม่มีความจำเป็น การใส่เป็นความฟุ่มเฟือย
- char myString[10] = {‘S’, ‘i’, ‘l’, ‘p’, ‘a’, ‘k’, ‘o’, ‘r’, ‘n’, ‘\0’};
  วิธีข้างบนนอกจากจะเยิ่นเย้อแล้วเรายังต้องออกแรงใส่อักขระศูนย์ด้วยตัวเอง
- ใส่ค่าอักขระเข้าไปทีละตัวก็ได้ แต่ฟุ่มเฟือยอย่างเป็นที่สุด

```
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’;
```


วิธีการประกาศสตริงที่ผิด

- ในขณะที่การประกาศที่ถูกมีหลายวิธี **วิธีที่ผิดก็มีไม่น้อยเหมือนกัน**
- ปัญหาที่พบบ่อย ๆ มีดังนี้
  - กำหนดขนาดอาเรย์ไว้ แต่ที่เก็บอักขระไม่พอ เช่น
    char myString[9] = “Silpakorn”; **ไม่เผื่อที่เก็บอักขระศูนย์เอาไว้**
  - สับสนเครื่องหมาย ‘ ’ กับ “ ” เช่น
    char myString[] = ‘Silpakorn’; **ที่จริงต้องใช้ “ ”**
    char myString[] = {“S”, “i”, “l”, “p”, “a”, “k”, “o”, “r”, “n”, “\0”};
    ที่จริงต้องใช้ ‘ ’
  - ลืมเครื่องหมาย เช่น
    char myString[] = Silpakorn;
    char myString[] = {Silpakorn};
  - ลืมใส่อักขระศูนย์ปิดท้าย
    char myString[] = {‘S’, ‘i’, ‘l’, ‘p’, ‘a’, ‘k’, ‘o’, ‘r’, ‘n’};

การรับและแสดงสตริงจากผู้ใช้

- เราใช้ scanf คู่กับ %s (ถ้าเป็นอักขระเราใช้ %c)
- พูดมาแค่นี้ดูเหมือนจะไม่มีอะไร แต่ที่จริงมันมีเรื่องเยอะมากเพราะว่า
  - เพราะสตริงเป็นอาเรย์ของอักขระ ดังนั้นเวลาใส่ค่าเข้าไป แท้จริงเราใส่ค่าเข้าไปในอาเรย์หลายช่องพร้อม ๆ กัน (อะไรมันจะขนาดนั้น)
  - ตัวแปรอาเรย์อยู่ในฐานะตัวชี้ (pointer) เวลารับค่าจาก scanf จะมีอะไรที่ผิดไปจากแบบอื่นที่เราพบมาก่อนหน้าอยู่บ้าง (คือไม่ต้องใช้ &)
- ตอนแสดงผลเราใช้ printf คู่กับ %s
  - ไม่ต้องกังวลว่ามันเป็นอาเรย์ ขอแค่มีอักขระศูนย์ปิดท้ายทุกอย่างจะดูดี
  - เราใส่ชื่อตัวแปรสตริงลงในใน printf เหมือนตัวแปรทั่วไป

ตัวอย่างโปรแกรมรับค่าสตริงจากผู้ใช้

```
char firstName[256];
char lastName[256];
```

สังเกตว่าเราใช้ firstName กับ lastName ใน scanf โดยไม่มี & เพราะสตริงเป็นตัวชี้อยู่แล้ว

```
printf("Enter your first name: ");
scanf("%s", firstName);
printf("Enter your last name: ");
scanf("%s", lastName);
```

เวลาใช้สตริงกับ printf ใส่ชื่อตัวแปรสตริงไปได้เลย แต่ให้ระวังว่าสตริงจะต้องมีอักขระศูนย์ปิดท้ายด้วย

```
printf("Your name is %s %s", firstName,
lastName);
```


การดำเนินการทางตัวเลขกับสตริง

เนื่องจากสตริงประกอบขึ้นมาจากอักขระ

- การดำเนินการทางตัวเลขที่ทำกับอักขระได้ก็จะทำบนสตริงได้ในลักษณะเดียวกัน
- แต่เราก็ต้องดึงเอาค่าอักขระออกมาประมวลผลทีละตัว
- การดึงอักขระออกมาทำได้เหมือนกับการอ่านค่าจากอาเรย์ทั่ว ๆ ไป เช่น
  - char c0 = str[0]; หมายถึงการดึงอักขระตัวแรกจาก str ไปเก็บไว้ที่ c0
  - str[3] = c0; หมายถึงการกำหนดให้อักขระตัวที่สี่ใน str มีค่าเท่ากับ c0
- หลังจากได้อักขระมาแล้ว เราสามารถดำเนินการบวกลบค่าตัวเลขกับอักขระหรือเปรียบเทียบค่าตัวเลขได้ในลักษณะเดียวกับที่เราทำกับอักขระทั่วไป

ตัวอย่างการดำเนินการทางตัวเลขกับสตริง

การประยุกต์ที่พบบ่อยในการดำเนินการทางตัวเลขกับสตริงก็คือ

- การเปลี่ยนอักขระในสตริงให้เป็นตัวใหญ่ให้หมดทุกตัว
- การเปลี่ยนอักขระในสตริงให้เป็นตัวเล็กให้หมดทุกตัว
- การเปลี่ยนอักขระที่ต้นประโยคให้เป็นตัวใหญ่ (ไมโครซอฟต์เวิร์ดก็ทำ)

หากเราไม่ใช้ฟังก์ชันไลบรารีมาตรฐาน วิธีการทั่วไปของเราก็คือเราต้องดำเนินการทางตัวเลขกับอักขระภายในสตริงทีละตัว

โจทย์ แปลงข้อความให้เป็นตัวพิมพ์ใหญ่

ตัวอย่าง จงเขียนโปรแกรมที่ทำการเปลี่ยนตัวอักษรทุกตัวในข้อความให้กลายเป็นตัวพิมพ์ใหญ่ โดยที่

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

ข้อมูลเข้า	ผลลัพธ์
Silpakorn	SILPAKORN
CP#	CP#

โปรแกรมแปลงข้อความให้เป็นตัวพิมพ์ใหญ่

```
scanf("%s", str);
```

เราดำเนินการแปลงตัวอักษรทีละตัว จนกว่าจะพบอักขระศูนย์ซึ่งเป็นตัวระบุจุดสิ้นสุดข้อความ

```
int i = 0;
while( str[i] != '\0' ) {
```

ตรวจดูว่าเป็นตัวพิมพ์เล็กหรือไม่

```
  if(str[i] >= 'a' && str[i] <= 'z') {
    str[i] -= 32; // Convert to uppercase
  }
  ++i;
}
printf("%s", str);
```


ข้อจำกัดของการใช้ scanf กับสตริง

- การสิ้นสุดของข้อความโดยทั่วไปแล้วเรานับจากการพบอักขระศูนย์
- แต่คำสั่ง scanf จะตัดข้อความเมื่อพบช่องว่าง ตัวกั้นหน้า และการขึ้นบรรทัดใหม่
  - แต่โดยทั่วไปเราอยากให้การรับข้อความจากผู้ใช้สิ้นสุดเมื่อขึ้นบรรทัดใหม่ (ผู้ใช้กดปุ่ม Enter)
  - เพื่อแก้ปัญหานี้เราใช้คำสั่งอื่นแทน scanf ในการรับข้อความ
  - คำสั่งที่ใช้ได้มีหลายอัน แต่อันที่เข้าใจง่ายที่สุด (แม้ไม่ดีนัก) คือ gets

```
char str[1024];
gets(str);
```


คำสั่ง gets

- คำสั่ง gets เป็นคำสั่งที่ใช้ในการรับข้อความจากผู้ใช้
  - ข้อความจะสิ้นสุดเมื่อพบการขึ้นบรรทัดใหม่
  - สามารถรับช่องว่างหรือตัวกั้นหน้าหลาย ๆ อันเก็บไว้ได้ในข้อความเดียว
- คำสั่ง gets สามารถใช้ได้ดังแสดงในตัวอย่างด้านใต้

```
char str[1024];
gets(str);
```

- เวลาเราใช้ gets เราไม่ต้องระบุ %s เหมือนกับที่เราทำใน scanf
  - เพราะตัว gets ใช้ได้แต่กับข้อมูลชนิดข้อความ
  - ส่วน scanf ใช้ได้กับข้อมูลหลายชนิดจึงต้องมีการระบุชนิดข้อมูลด้วย

ตัวอย่างการใช้ gets แก้ปัญหาช่องว่างในข้อมูลเข้า

จงเขียนโปรแกรมที่ทำการเปลี่ยนตัวอักษรทุกตัวในข้อความให้กลายเป็นตัวพิมพ์ใหญ่ โดยที่ข้อความได้มาจากผู้ใช้และอาจมีช่องว่างหรือตัวกั้นหน้ารวมอยู่ด้วย กำหนดให้ข้อความจากผู้ใช้สิ้นสุดเมื่อผู้ใช้สั่งขึ้นบรรทัดใหม่

ตัวอย่างข้อมูลเข้าและผลลัพธ์

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

โค้ดตัวอย่างการใช้ gets แก้ปัญหาช่องว่างในข้อมูลเข้า

```
char str[1024];
```

เปลี่ยนจาก scanf เป็น gets ที่เดียวก็พอแล้ว

```
gets(str);

int i = 0;
while(str[i] != '\0') {
  if(str[i] >= 97 && str[i] <= 122) {
    str[i] -= 32;   // Convert to uppercase
  }
  ++i;
}
printf("%s", str);
```


หัวข้อเนื้อหา

- ฟังก์ชันไลบรารีมาตรฐานคืออะไร
- Header File กับฟังก์ชันไลบรารีมาตรฐาน
- ฟังก์ชันคณิตศาสตร์
- ข้อมูลชนิดตัวอักขระ (character)
- ฟังก์ชันเกี่ยวกับตัวอักขระ
- สตริง
- **ฟังก์ชันไลบรารีมาตรฐานสำหรับจัดการสตริง**

ฟังก์ชันไลบรารีมาตรฐานสำหรับจัดการสตริง

- สามารถเรียกใช้ได้จาก string.h เช่น #include <string.h>
- มีฟังก์ชันที่มีประโยชน์มากอยู่หลายตัว เช่น
  - การหาความยาวของข้อความ
  - การเปรียบเทียบข้อความ
  - การนำสตริงสองอันมาต่อกัน
- ในทางทฤษฎี เราสามารถเขียนฟังก์ชันเหล่านี้ด้วยตัวเองได้
  - แต่ในเมื่อมีคนทำมาไว้ให้เราแล้ว เราก็ไม่จำเป็นต้องเขียนใหม่
  - โค้ดใน string.h มักจะทำงานเร็วกว่าที่เราเขียนเอง
  - แทบจะเป็นไปไม่ได้เลยที่จะฟังก์ชันใน string.h จะมีข้อผิดพลาด

strlen คำสั่งหาความยาวสตริง

ตัวอย่างการใช้งาน

```
char myString[256] = "Silpakorn";
int length = strlen(myString);
printf("Length = %d", length);
```

- ตัวอย่างข้างบนจะพิมพ์เลข 9 ซึ่งคือความยาวของสตริงออกมา
- ฟังก์ชันไม่นับอักขระศูนย์ (ซึ่งโดยปรกติเราก็ไม่อยากนับอยู่แล้ว)
- เรานิยมใช้ strlen เพื่อใช้ในการหาจำนวนตัวอักขระและระบุจำนวนรอบในการวนลูปที่ควรจะเป็น
- **สังเกตด้วยว่าเราใช้ชื่อตัวแปรตรง ๆ เลย เพราะฟังก์ชันใน string.h รับส่งสตริงในรูปตัวชี้ตลอด**

strcat คำสั่งต่อสตริงสองอันเข้าด้วยกัน

ตัวอย่างการใช้งาน

```
char str[256] = "Silpakorn";
char str2[256] = "University";
char* outString = strcat(str, str2);
printf("%s", outString);
```

- สำหรับตัวอย่างข้างบนเราจะได้ผลลัพธ์เป็น SilpakornUniversity
- strcat ไม่ใส่ช่องว่างให้ ถ้าอยากได้ช่องว่างต้องหาทางใส่เอง
- บางคำสั่งเราจำเป็นต้องใช้รูปตัวชี้อย่างชัดเจนเพื่อรับผลลัพธ์

strcmp คำสั่งเปรียบเทียบสตริง

เป็นคำสั่งเปรียบเทียบอักขระแต่ละตัวของสตริงจากซ้ายไปขวาทีละตัว

- รูปแบบคำสั่งคือ int result = strcmp(char* string1, char* string2);
- การเปรียบเทียบจะทำจากซ้ายไปขวาทีละตัวอักษรจนกว่าจะพบความแตกต่างของตัวอักษรใน string1 และ string2 หรือสิ้นสุดข้อความ
- ความแตกต่างวัดกันที่รหัสแอสกี ดังนั้นตัวพิมพ์ใหญ่และเล็กจะให้ความแตกต่างในการเปรียบเทียบ (case sensitive)

เมื่อพบความแตกต่างของอักขระในสตริงทั้งสอง สิ่งที่เกิดขึ้นตามมาจะเป็นดังนี้

- ถ้ารหัสแอสกีของอักขระดังกล่าวใน string1 น้อยกว่า string2 ฟังก์ชันจะคืนค่าลบ และถ้ามากกว่าจะได้ค่าบวก
- ถ้าพบอักขระศูนย์ใน string1 ก่อน string2 จะคืนค่าลบ
- ถ้าถ้าพบอักขระศูนย์ใน string2 ก่อน string1 จะคืนค่าบวก

ถ้าสตริงทั้งสองไม่มีความแตกต่างใด ๆ เลย ฟังก์ชันจะคืนค่าศูนย์กลับมาให้

ตัวอย่างการใช้ strcmp (1)

strcmp(“Hello world”, “hello World”);

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

strcmp(“hello world”, “hello World”);

- แบบนี้จะได้ค่าเป็นบวก เพราะการเปรียบเทียบจะดำเนินไปทีละตัวจนกว่าจะพบความแตกต่าง ซึ่งอยู่ที่ตัวอักษร W
- อักขระใน string1 เป็นตัวพิมพ์เล็กจึงมีค่าแอสกีมากกว่า ผลลัพธ์จึงเป็นบวก

ตัวอย่างการใช้ strcmp (2)

strcmp(“Hello World”, “Hello World”);
ได้ผลลัพธ์เป็นศูนย์ เพราะสตริงเหมือนกันทุกประการ

strcmp(“Hello”, “Hello World”);
ได้ผลลัพธ์เป็นลบเพราะ string1 หมดก่อนที่จะพบความแตกต่างในอักขระอื่น ๆ

strcmp(“BOOT”, “BUT”);

- ได้ผลลัพธ์เป็นลบเนื่องจากอักขระที่พบความแตกต่างตัวแรกคือตัวที่สองและ O มีรหัสแอสกีน้อยกว่า U
- ความยาวของสตริงไม่ใช่ตัวตัดสิน แต่เป็นอักขระแรกที่พบความแตกต่าง

strcmp(“Boot”, “BUt”);

- ได้ค่าเป็นบวกเพราะอักขระแรกที่พบความแตกต่างคือตัวที่สอง
- แต่ o เป็นตัวพิมพ์เล็กซึ่งมีค่ารหัสแอสกีมากกว่าตัวพิมพ์ใหญ่ ดังนั้นการเปรียบเทียบจึงได้ผลลัพธ์เป็นบวก

strcmpi เปรียบเทียบสตริงตามหลักพจนานุกรม

- คำสั่ง strcmp ใช้การเปรียบเทียบรหัสแอสกีทำให้ตัวพิมพ์ใหญ่และตัวพิมพ์เล็กถูกพิจารณาว่าเป็นคนละตัวอักษร
- ทำให้ตัว Z มาก่อนตัว a ซึ่งขัดกับหลักพจนานุกรม
- เราแก้ปัญหานี้ได้ด้วยการใช้คำสั่ง
  int strcmpi (char* string1, char* string2)
  - ตัวอักษรใน string1 และ string2 จะถูกมองว่ามีรหัสแอสกีเหมือนตัวพิมพ์เล็ก
  - ทำให้การเปรียบเทียบตัวอักษรไม่มีความต่างกันระหว่างตัวพิมพ์ใหญ่และเล็ก
  - ใช้รหัสแอสกีของตัวพิมพ์เล็กในการเปรียบเทียบตัวอักษรอังกฤษกับเครื่องหมายวรรคตอนหรือสัญลักษณ์พิเศษ

ตัวอย่างการใช้ strcmpi

- **strcmpi(“Hello world”, “hello World”);**
  strcmpi(“hello world”, “hello World”);
  strcmpi(“Hello World”, “Hello World”);
  การเปรียบเทียบทั้งสามอันข้างต้นจะได้ค่าเป็นศูนย์หมดเพราะตัวใหญ่ตัวเล็กไม่มีความหมายกับฟังก์ชัน strcmpi
- **strcmpi(“Boot”, “BUt”);**
  ได้ค่าเป็นลบเพราะคำว่า boot มาก่อน but ตามหลักพจนานุกรม
- **strcmpi(“bu^”, “BUT”);**
  ได้ค่าเป็นลบเพราะว่า ^ มาก่อนตัว t (ค่าแอสกีที่ใช้เปรียบเทียบมาจากตัวพิมพ์เล็ก)
- **strcmpi(“bu{“, “BUT”);**
  ได้ค่าเป็นบวก เพราะ { มีค่า { มากกว่า t

สรุปเกี่ยวกับสตริง

- อักขระและสตริงเป็นของคนละอย่างแต่สตริงสร้างขึ้นมาจากการนำอักขระมาต่อกันด้วยอาเรย์
- อักขระแต่ละตัวแท้จริงแล้วมีตัวเลขกำกับอยู่ตามมาตรฐานแอสกี
- เราสามารถดำเนินการทางตัวเลขกับอักขระและสตริงได้
- สตริงที่ดีควรจะมีอักขระศูนย์ปิดท้ายเสมอ
- ดังนั้นเราต้องเผื่อที่ในอาเรย์ไว้เก็บอักขระศูนย์ในสตริงด้วย
- มีคำสั่งอรรถประโยชน์มากมายใน string.h

สรุปเกี่ยวกับฟังก์ชันไลบรารีมาตรฐาน

- เราเรียกใช้ฟังก์ชันไลบรารีมาตรฐานได้จากเฮดเดอร์ไฟล์ต่าง ๆ
- ฟังก์ชันไลบรารีมาตรฐานถูกจัดเป็นหมวดหมู่ตามเฮดเดอร์ไฟล์
- ในหมวดคณิตศาสตร์ มีฟังก์ชันจำนวนมากที่การบวกลบคูณหารปรกติไม่สามารถทำให้เราหาค่าออกมาได้ในเวลาอันสั้น
- ในหมวดตัวอักขระและสตริง ส่วนใหญ่แล้วเราไม่จำเป็นต้องใช้เพราะการบวกลบคูณหารและการเปรียบเทียบอักขระทั่วไป สามารถทำหน้าที่ในฟังก์ชันมาตรฐานได้
  - แต่การใช้ฟังก์ชันไลบรารีมาตรฐานมักจะทำให้โปรแกรมเร็วกว่า และมีความผิดพลาดน้อยกว่า
  - ถ้าใช้ฟังก์ชันไลบรารีมาตรฐานเราก็ไม่จำเป็นต้องออกแรงเขียนฟังก์ชันเอง