



# เขียนโปรแกรมภาษา C

## สำหรับผู้เริ่มต้น [Phase2]

# ฟังก์ชัน getchar และ putchar

การรับและแสดงผลข้อมูลแบบตัวอักษรนอกจากจะใช้ฟังก์ชัน `scanf()` และ `printf()` แล้ว ยังมีฟังก์ชันเฉพาะที่ใช้สำหรับรับข้อมูลแบบตัวอักษร คือ

- **getchar()** คือ ฟังก์ชันสำหรับรับข้อมูล 1 ตัวอักษรจากคีย์บอร์ด
- **putchar()** คือ ฟังก์ชันสำหรับแสดงผลข้อมูล 1 ตัวอักษรออกทางจอภาพ

# ฟังก์ชัน gets และ puts

การรับและแสดงผลชุดข้อความ (String) นอกจากจะใช้ฟังก์ชัน scanf() และ printf() แล้ว ยังมีฟังก์ชันเฉพาะที่ใช้จัดการข้อความ คือ

- **gets()** คือ ฟังก์ชันสำหรับรับข้อมูลชุดข้อความจากคีย์บอร์ด
- **puts()** คือ ฟังก์ชันสำหรับแสดงผลชุดข้อความออกทางจอภาพ

รู้จักกับอาร์เรย์

# ข้อจำกัดของชนิดข้อมูลพื้นฐาน

การประกาศตัวแปรแต่ละครั้ง ตัวแปร 1 ตัวสามารถเก็บข้อมูลได้แค่ 1 ค่าเท่านั้น เช่น

```
int score1 = 100
```

```
int score2 = 80
```

```
int score3 = 65
```

# ตัวอย่างการสร้างตัวแปรแบบปกติ

```
int score1 = 100
```

```
int score2 = 80
```

```
int score3 = 65
```

score1



100

score2



80

score3



65

# ข้อจำกัดของชนิดข้อมูลพื้นฐาน

“ ถ้าอยากเก็บเลข 10 ค่าต้องทำอะไร ต้องประกาศ  
ตัวแปรจำนวน 10 ตัวแปร หรือไม่ ? ”



# ตัวอย่างการสร้างตัวแปรแบบปกติ

```
int score1 = 100
```

```
int score2 = 80
```

```
int score3 = 65
```

```
int score4 = 80
```

```
int score5 = 90
```

```
int scoreN = xx
```

score1



100

score2



80

score3



65



# อาร์เรย์ คืออะไร

1. ชุดของตัวแปรที่อยู่ในรูปลำดับใช้เก็บค่าข้อมูลให้อยู่ในกลุ่มเดียวกัน โดยข้อมูลภายในอาร์เรย์จะถูกเก็บในตำแหน่งที่ต่อเนื่องกัน
2. เป็นตัวแปรที่ใช้ในการเก็บข้อมูลที่มีลำดับที่ต่อเนื่อง ซึ่งข้อมูลมีค่าได้หลายค่าโดยใช้ชื่ออ้างอิงได้เพียงชื่อเดียว และใช้หมายเลขกำกับ (**index**) ให้กับตัวแปรเพื่อจำแนกความแตกต่างของค่าตัวแปรแต่ละตัว

# คุณสมบัติของอาร์เรย์

1. ใช้เก็บกลุ่มของข้อมูล
2. ข้อมูลที่อยู่ในอาร์เรย์จะเรียกว่าสมาชิก หรือ อิลิเมนต์ (element)
3. แต่ละอิลิเมนต์ (element) จะเก็บค่าข้อมูล (value) และ อินเด็กซ์ (Index)
4. Index หมายถึงคีย์ของอาร์เรย์ใช้อ้างอิงตำแหน่งของ element เริ่มต้นที่ 0
5. สมาชิกในอาร์เรย์ต้องมีชนิดข้อมูลเหมือนกัน
6. สมาชิกในอาร์เรย์จะถูกค้นด้วยเครื่องหมายคอมม่า



# ตัวอย่างการสร้างตัวแปรแบบปกติ

```
int score1 = 100
```

```
int score2 = 80
```

```
int score3 = 65
```

score1



100

score2



80

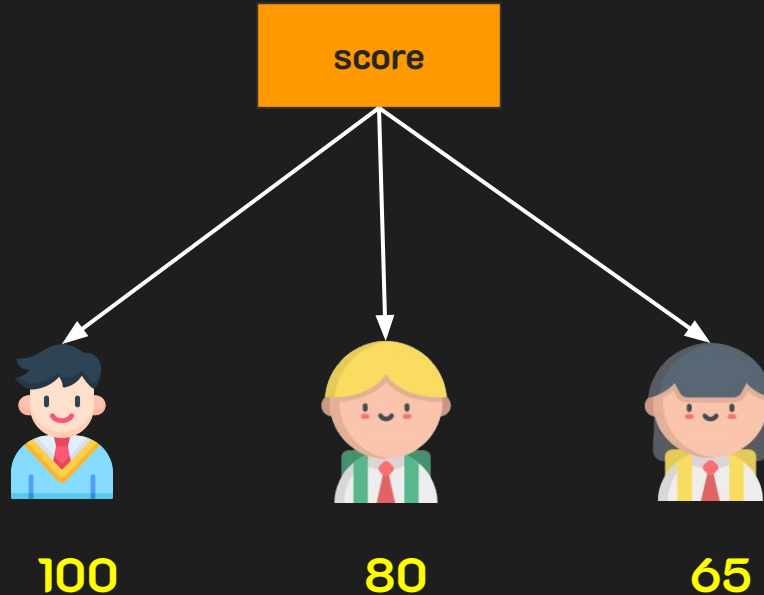
score3



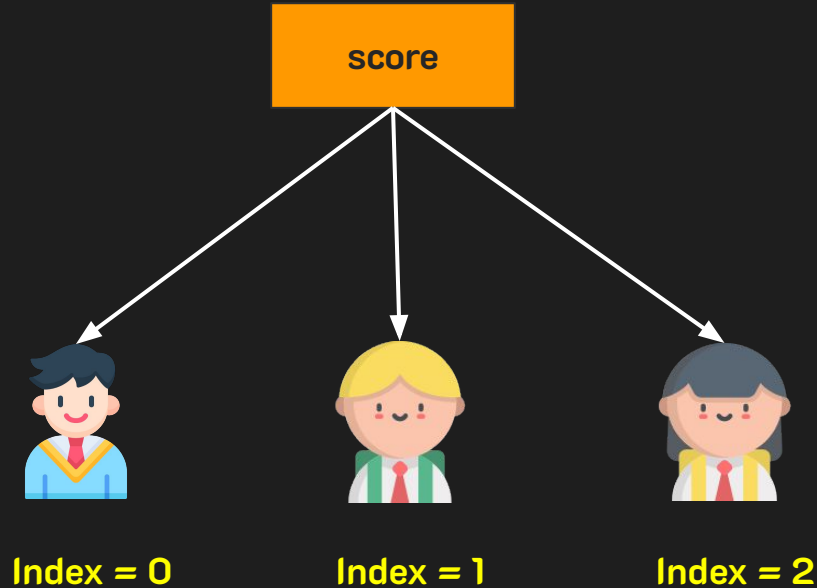
65



# ตัวอย่างการสร้างตัวแปรอาร์เรย์



# เข้าถึงสมาชิกในตัวแปรอาร์เรย์



# สรุปอาร์เรย์

1. ใช้เก็บกลุ่มของข้อมูล ที่มีชนิดข้อมูลเดียวกัน
2. ใช้ตัวแปรชื่อเดียวกัน
3. ใช้หมายเลขกำกับเพื่ออ้างอิงตำแหน่งของข้อมูลในอาร์เรย์
4. มีขนาดที่แน่นอนไม่สามารถปรับเปลี่ยนขนาดได้



# การสร้างอาร์เรย์ (Array)

# การสร้างอาร์เรย์

## แบบกำหนดขนาด

ชนิดข้อมูล ชื่อตัวแปร[ขนาด]; **//ขนาดต้องเป็นตัวเลขจำนวนเต็ม**  
เช่น `int score [3];`

## แบบกำหนดขนาดและค่าเริ่มต้น

ชนิดข้อมูล ชื่อตัวแปร [ขนาด] = {สมาชิก,...};  
เช่น `int score [3] = {100,90,70};`



# การสร้างอาร์เรย์

```
int score [4] = {100,90,70,80};
```

100	90	70	80
-----	----	----	----

```
int score [10] = {100,90,70,80};
```

100	90	70	80	0	0	0	0	0	0
-----	----	----	----	---	---	---	---	---	---

# การสร้างอาร์เรย์

```
int score [4] = {100,90,70,80};
```

100	90	70	80
-----	----	----	----

```
int score [10] = {100,90,70,80};
```

100	90	70	80	0	0	0	0	0	0
-----	----	----	----	---	---	---	---	---	---

# การสร้างอาร์เรย์

## แบบไม่กำหนดขนาด

ชนิดข้อมูล ชื่อตัวแปร[] = {สมาชิก,...};  
เช่น `int score [] = {100,90,70,80};`

100

90

70

80

# จัดการสมาชิกใน อาร์เรย์

# การเข้าถึงสมาชิก

```
int score [3] = {100, 90, 80};
```

100	90	80
-----	----	----

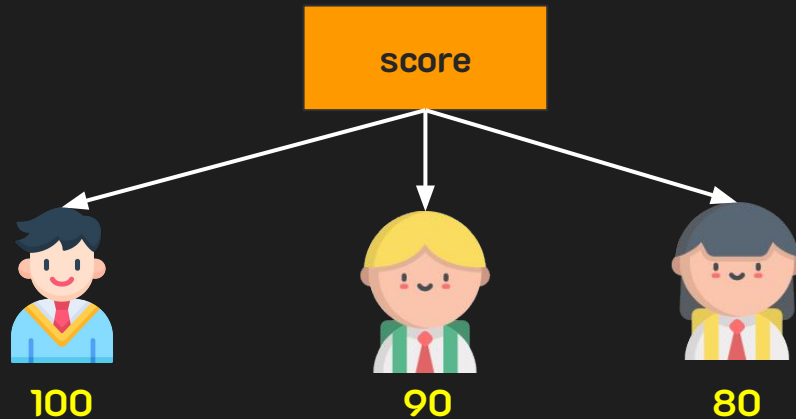
# การเข้าถึงสมาชิก

```
int score [3] = {100, 90, 80};
```

100 (0)	90 (1)	80 (2)
---------	--------	--------

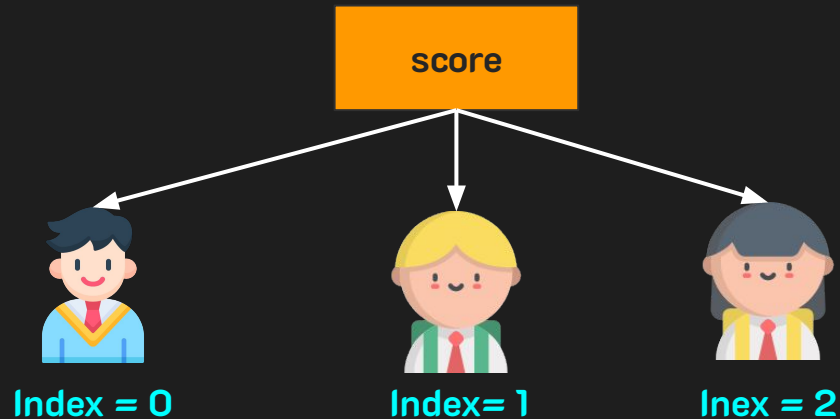
# การเข้าถึงสมาชิก

```
int score [3] = {100, 90, 80};
```



# การเข้าถึงสมาชิก

```
int score [3] = {100,90,80};
```





# การเปลี่ยนแปลงข้อมูลสมาชิก Array

```
int number[] = {10, 20, 30, 40};
```

```
number[2] = 100;
```

```
char vowels[] = {'A','F','I','O','U'}
```

```
vowels[1]='E';
```

# การเข้าถึงสมาชิกด้วย For Loop

```
int number[] = {10, 20, 30};  
for (int i = 0; i < 3; i++) {  
    // กระบวนการทำงาน  
}
```

อาร์เรย์ 2 มิติ

# อาร์เรย์ 2 มิติ

- อาร์เรย์ที่มีข้อมูลสมาชิกภายในเป็นอาร์เรย์ (Array ซ้อน Array)
- มีโครงสร้างเป็นรูปแบบแถว (แนวนอน) และคอลัมน์ (แนวตั้ง)



# รูปแบบของอาร์เรย์ 1 มิติ

```
int number [] = {10, 20, 30, 40};
```

10	20	30	40
----	----	----	----



Array 1 มิติ

# รูปแบบของอาร์เรย์ 2 มิติ

แถวที่ 0

--	--	--	--

แถวที่ 1

--	--	--	--

แถวที่ 2

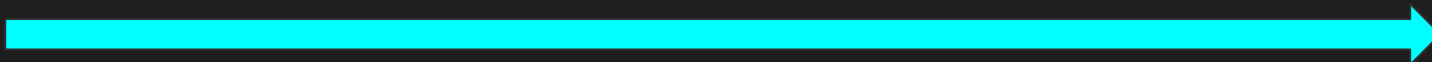
--	--	--	--

# รูปแบบของอาร์เรย์ 2 มิติ

แถวที่ 0

แถวที่ 1

แถวที่ 2

# รูปแบบของอาร์เรย์ 2 มิติ

	คอลัมน์ที่ 0	คอลัมน์ที่ 1	คอลัมน์ที่ 2	คอลัมน์ที่ 3
แถวที่ 0				
แถวที่ 1				
แถวที่ 2				





# การเข้าถึงสมาชิกในอาร์เรย์ 2 มิติ

	คอลัมน์ที่ 0	คอลัมน์ที่ 1	คอลัมน์ที่ 2	คอลัมน์ที่ 3
แถวที่ 0	[0,0]	[0,1]	[0,2]	[0,3]
แถวที่ 1	[1,0]	[1,1]	[1,2]	[1,3]
แถวที่ 2	[2,0]	[2,1]	[2,2]	[2,3]



สร้างอาร์เรย์ 2 มิติ

# การสร้างอาร์เรย์ 2 มิติ

## แบบกำหนดขนาด

ชนิดข้อมูล ชื่อตัวแปร[ขนาดแถว][ขนาดคอลัมน์];  
เช่น `int score [2][4];`

# การสร้างอาร์เรย์ 2 มิติ

## แบบกำหนดขนาดและค่าเริ่มต้น

ชนิดข้อมูล ชื่อตัวแปร[ขนาดแถว][ขนาดคอลัมน์] = {สมาชิก,...};

ตัวอย่าง เช่น

```
int numbers [2][4]={  
    {50,70,80,90},  
    {100,99,60,55}  
};
```

# การสร้างอาร์เรย์ 2 มิติ

## แบบกำหนดขนาดและค่าเริ่มต้น

ชนิดข้อมูล ชื่อตัวแปร[ขนาดแถว][ขนาดคอลัมน์] = {สมาชิก,...};

ตัวอย่าง เช่น

```
int numbers [2][4]={
```

แถว 1

{50,70,80,90},

{100,99,60,55}

```
};
```

# การสร้างอาร์เรย์ 2 มิติ

## แบบกำหนดขนาดและค่าเริ่มต้น

ชนิดข้อมูล ชื่อตัวแปร[ขนาดแถว][ขนาดคอลัมน์] = {สมาชิก,...};

ตัวอย่าง เช่น

```
int numbers [2][4]={
```

แถว 1

{50,70,80,90},

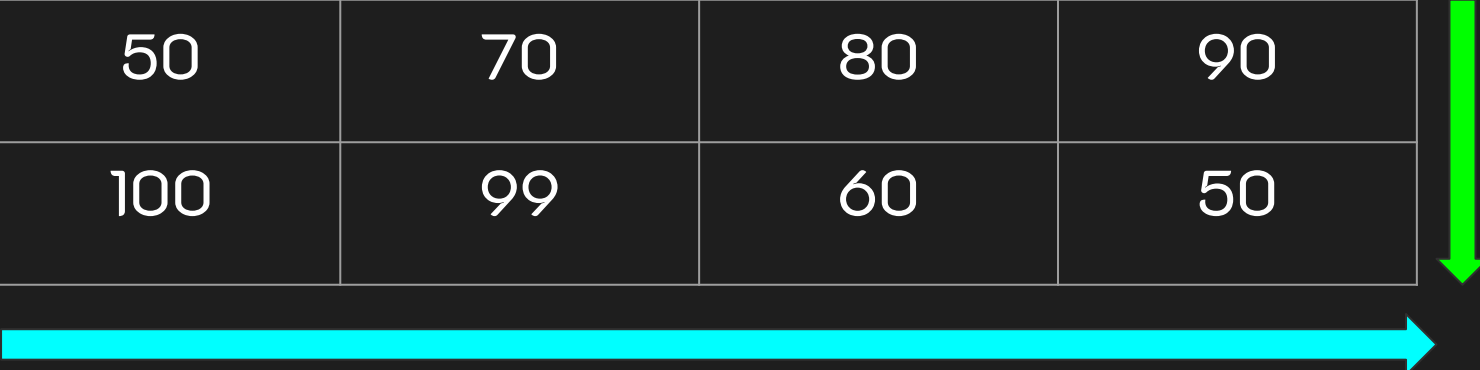
แถว 2

{100,99,60,55}

```
};
```

# โครงสร้างของอาร์เรย์ 2 มิติ

	คอลัมน์ที่ 0	คอลัมน์ที่ 1	คอลัมน์ที่ 2	คอลัมน์ที่ 3
แถวที่ 0	50	70	80	90
แถวที่ 1	100	99	60	50



# การเข้าถึงสมาชิกอาร์เรย์ 2 มิติ

	คอลัมน์ที่ 0	คอลัมน์ที่ 1	คอลัมน์ที่ 2	คอลัมน์ที่ 3
แถวที่ 0	50 [0,0]	70 [0,1]	80 [0,2]	90 [0,3]
แถวที่ 1	100 [1,0]	99 [1,1]	60 [1,2]	50 [1,3]





# การเปลี่ยนแปลงค่าในอาร์เรย์ 2 มิติ

## โครงสร้างคำสั่ง

ชื่อตัวแปร[แถว][คอลัมน์] = กำหนดค่า

score [0][1] = 99

score [1][3] = 80

ฟังก์ชัน (Function)

# ฟังก์ชัน (Function) คืออะไร

ชุดคำสั่งที่นำมาเขียนรวมกันเป็นกลุ่มเพื่อให้เรียกใช้งานตาม  
วัตถุประสงค์ที่ต้องการและลดความซ้ำซ้อนของคำสั่งที่ใช้งานบ่อย

ฟังก์ชันสามารถนำไปใช้งานได้ทุกที่และแก้ไขได้ในภายหลัง  
ทำให้โค้ดในโปรแกรมมีระเบียบและใช้งานได้สะดวกมากยิ่งขึ้น

# ประเภทของฟังก์ชัน

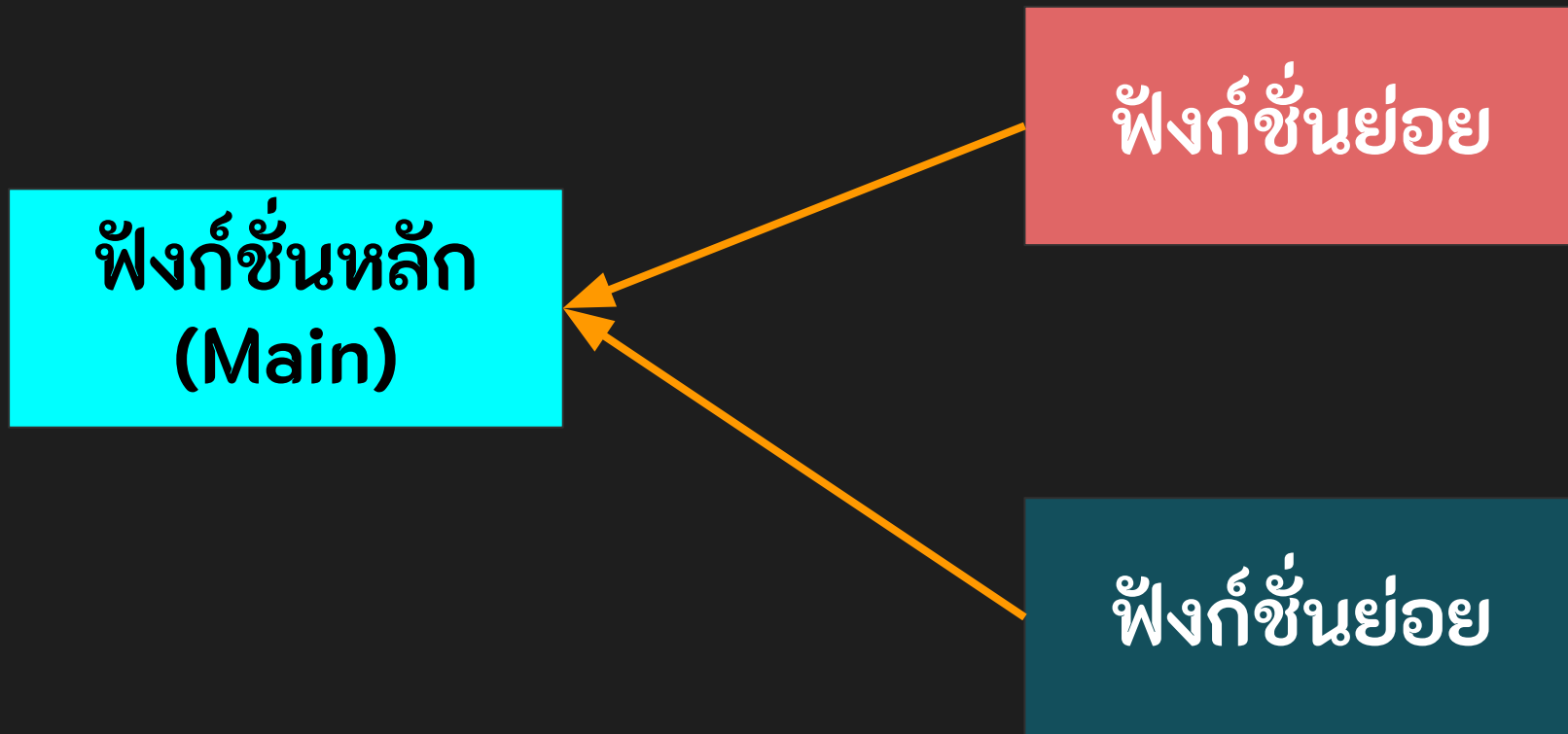
- **ฟังก์ชันมาตรฐาน (Standard Library Functions)** คือ ฟังก์ชันที่มีอยู่ในภาษา C ผู้ใช้สามารถเรียกใช้งานได้เลย เช่น printf() , scanf() ที่ทำงานอยู่ในไลบรารี หรือ Header File เช่น stdio.h เป็นต้น
- **ฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง (User-Define Function)** คือ ฟังก์ชันที่ถูกสร้างขึ้นมาเพื่อวัตถุประสงค์ให้ทำงานตามที่ใช้ต้องการ

# ฟังก์ชันหลัก (main)

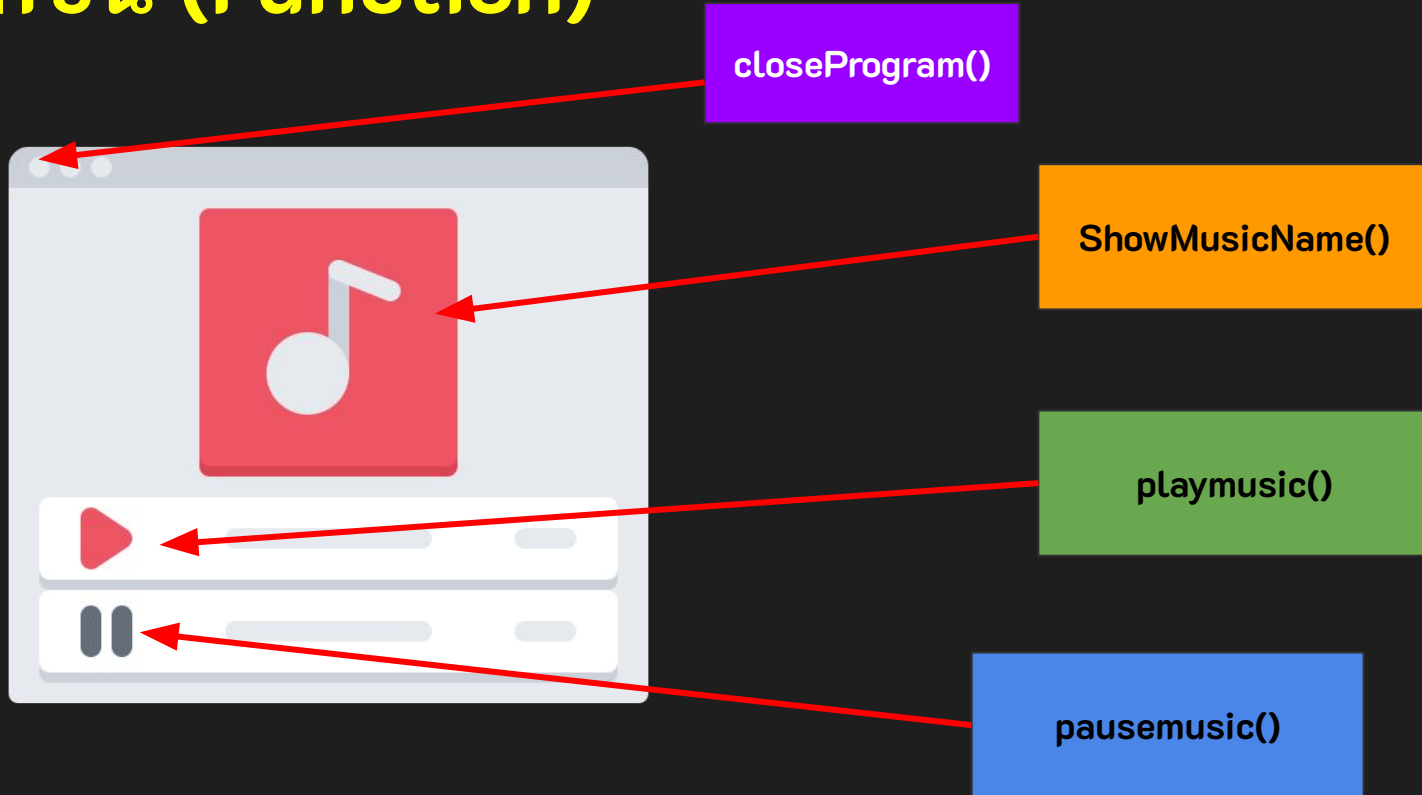
```
int main()  
{  
  
    //statement  
  
}
```

ฟังก์ชัน **main()** คือ ฟังก์ชันพิเศษกลุ่มคำสั่งที่อยู่ในฟังก์ชันนี้จะถูกสั่งให้ทำงานโดยอัตโนมัติเป็นลำดับแรกเสมอ

# ฟังก์ชัน (Function)



# ฟังก์ชัน (Function)



# รูปแบบฟังก์ชัน

- ฟังก์ชันแบบปกติ (Void Function)
- ฟังก์ชันแบบมีพารามิเตอร์ (Parameter Function)
- ฟังก์ชันแบบมีค่าส่งกลับ (Return Function)
- ฟังก์ชันแบบรับและส่งค่า



# กฎการตั้งชื่อฟังก์ชัน

- ชื่อฟังก์ชันต้องไม่ซ้ำกัน
- ชื่อฟังก์ชันสามารถตั้งเป็นตัวอักษรหรือตัวเลขได้
- ชื่อของฟังก์ชันต้องไม่ขึ้นต้นด้วยตัวเลข

# วิธีสร้างฟังก์ชัน

## 1. นิยามชื่อฟังก์ชันก่อนกำหนดโครงสร้าง

```
func_name(); //นิยามชื่อฟังก์ชัน (function prototype)
```

```
int main(){  
}
```

```
func_name(){} //กำหนดโครงสร้างการทำงานหลังฟังก์ชัน main
```

# วิธีสร้างฟังก์ชัน

## 2. นิยามชื่อฟังก์ชันพร้อมกำหนดโครงสร้าง

// นิยามชื่อพร้อมกำหนดโครงสร้างคำสั่ง (เขียนอยู่ด้านบน main เท่านั้น)

```
func_name(){ }
```

```
int main(){  
}
```

ฟังก็ชื่นแบบปกติ

# ฟังก์ชันที่ไม่มีการรับและส่งค่า (void)

## โครงสร้างคำสั่ง

```
void ชื่อฟังก์ชัน(){  
    // คำสั่งต่างๆ  
}
```

## การเรียกใช้งานฟังก์ชัน

```
ชื่อฟังก์ชัน ();
```

ฟังก์ชันแบบมีพารามิเตอร์

# ฟังก์ชันแบบมีพารามิเตอร์ (Parameter)

## โครงสร้างคำสั่ง

```
void ชื่อฟังก์ชัน(parameter1,parameter2,...){  
    // กลุ่มคำสั่งต่างๆ  
}
```

## การเรียกใช้งานฟังก์ชัน

```
ชื่อฟังก์ชัน (argument1,argument2,...);
```

# ฟังก์ชันแบบมีพารามิเตอร์ (Parameter)

## โครงสร้างคำสั่ง

```
void ชื่อฟังก์ชัน(parameter1,parameter2,...){  
    // กลุ่มคำสั่งต่างๆ  
}
```

## การเรียกใช้งานฟังก์ชัน

ชื่อฟังก์ชัน (argument1,argument2,...);

- อาร์กิวเมนต์ คือ ตัวแปรหรือค่าที่ต้องการส่งมาให้กับฟังก์ชัน (ตัวแปรส่ง)
- พารามิเตอร์ คือ ตัวแปรที่ฟังก์ชันสร้างไว้สำหรับรับค่าที่จะส่งเข้ามาให้กับฟังก์ชัน (ตัวแปรรับ)



ฟังก็ชื่นแบบมีค่าส่งกลับ

# ฟังก์ชันแบบมีค่าส่งกลับ (Return)

## โครงสร้างคำสั่ง

```
type ชื่อฟังก์ชัน(){  
    return ค่าที่จะส่งออกไป (อ้างอิงตามชนิดข้อมูล)  
}
```

## การเรียกใช้งานฟังก์ชัน

ตัวแปรที่รับค่าจากฟังก์ชัน = ชื่อฟังก์ชัน ();

ฟังก์ชันแบบรับและส่งค่า

# ฟังก์ชันแบบรับและส่งค่า

## โครงสร้างคำสั่ง

```
type ชื่อฟังก์ชัน(parameter1,parameter2,...){  
    return ค่าที่จะส่งออกไป (อ้างอิงตามชนิดข้อมูล)  
}
```

## การเรียกใช้งานฟังก์ชัน

ตัวแปรที่รับค่าจากฟังก์ชัน = ชื่อฟังก์ชัน(argument1,argument2..);

ขอบเขตตัวแปร

# ขอบเขตตัวแปร

- **Local variable** ตัวแปรที่ประกาศอยู่ภายในฟังก์ชันมีขอบเขตการทำงานตั้งแต่จุดเริ่มต้นไปจนถึงจุดสิ้นสุดของฟังก์ชันจะถือได้ว่าฟังก์ชันนั้นเป็นเจ้าของตัวแปรนั้น ฟังก์ชันอื่นจะไม่สามารถเรียกใช้งานตัวแปรนี้ได้

# ขอบเขตตัวแปร

- **Global variable** ตัวแปรที่ประกาศอยู่นอกฟังก์ชันมีขอบเขตการทำงานตั้งแต่จุดเริ่มต้นไปจนถึงจุดสิ้นสุดของไฟล์ที่ประกาศใช้ นั่นหมายถึงตัวแปรดังกล่าวนี้เป็นสาธารณะ ไม่มีฟังก์ชันใดเป็นเจ้าของ ทุกฟังก์ชันสามารถเรียกใช้งานตัวแปรนี้ได้

รู้จักกับ Pointer



# การสร้างตัวแปร

ตัวอย่าง

```
int number = 100;
```

ตัวแปร `number` เก็บค่าตัวเลขจำนวนเต็มคือเลข 100



<https://www.youtube.com/c/KongRuksiamTutorial>



<https://www.facebook.com/KongRuksiamTutorial/>

# การสร้างตัวแปร

ตัวอย่าง

```
int number = 100;
```

ตัวแปร `number` เก็บค่าตัวเลขจำนวนเต็มคือเลข 100

คำถาม : ตัวแปร `number` และค่า 100 ถูกเก็บไว้ที่ใด ???

# การสร้างตัวแปร

ตัวอย่าง

```
int number = 100;
```

ตัวแปร number เก็บค่าตัวเลขจำนวนเต็มคือเลข 100

คำถาม : ตัวแปร number และค่า 100 ถูกเก็บไว้ที่ใด ???

คำตอบ : เก็บไว้ที่หน่วยความจำ

# การสร้างตัวแปร

ตัวอย่าง

```
int number = 100;
```

ตัวแปร number เก็บค่าตัวเลขจำนวนเต็มคือเลข 100

คำถาม : แล้วจะเข้าถึงค่าในหน่วยความจำได้อย่างไร ??

# การสร้างตัวแปร

ตัวอย่าง

```
int number = 100;
```

ตัวแปร number เก็บค่าตัวเลขจำนวนเต็มคือเลข 100

คำถาม : แล้วจะเข้าถึงค่าในหน่วยความจำได้อย่างไร ??

คำตอบ : ใช้พอยน์เตอร์ (Pointer)

# Pointer คืออะไร

พอยน์เตอร์ (Pointer) คือตัวแปรที่ใช้เก็บ**ตำแหน่งที่อยู่**ของตัวแปรที่สนใจหรือค่าแอดเดรส (Address) หน่วยความจำ ซึ่งมีประโยชน์อย่างมากสำหรับการเขียนโปรแกรมจัดการหน่วยความจำ

# โครงสร้างพื้นที่หน่วยความจำ

ตัวแปร	ค่าในตัวแปร	แอดเดรส (Address)
a	10	0x6ffe3c
b	20	0x6ffe38
c	'A'	0x6ffe30

# โครงสร้างพื้นที่หน่วยความจำ

ตัวแปร	ค่าในตัวแปร	แอดเดรส (Address)
a	10	0x6ffe3c
b	20	0x6ffe38
c	'A'	0x6ffe30

**Address คือ ตำแหน่งที่เก็บข้อมูลในหน่วยความจำ เป็นรูปแบบเลขฐาน 16**



# การสร้างตัวแปร Pointer

# การสร้างตัวแปร Pointer

โครงสร้างคำสั่ง

ชนิดข้อมูล \*ตัวแปรพอยน์เตอร์;

# การสร้างตัวแปร Pointer

## ตัวอย่าง

```
int number = 10;
```

```
int *p1 = &number;
```

```
char letter = 'A';
```

```
char *p2 = &letter;
```

# การสร้างตัวแปร Pointer

## ตัวอย่าง

```
int number = 10;
```

```
int *p1 = &number;
```

```
char letter = 'A';
```

```
char *p2 = &letter;
```

# การสร้างตัวแปร Pointer

## ตัวอย่าง

```
int number = 10;
```

```
int *p1 = &number;
```

```
char letter = 'A';
```

```
char *p2 = &letter;
```

\* คือ ตำแหน่งแอดเดรสในหน่วย  
ความจำที่พอยน์เตอร์ชี้อยู่

& คือ ค่าแอดเดรสของตัวแปร

# การสร้างตัวแปร Pointer

## ตัวอย่าง

```
int number = 10;
```

```
int *p1 = &number;
```

```
char letter = 'A';
```

```
char *p2 = &letter;
```

ตัวแปร p1 คือ ตัวแปร pointer ที่ชี้ไปที่แอดเดรสของตัวแปรที่เป็นรูปแบบ int

ตัวแปร p2 คือ ตัวแปร pointer ที่ชี้ไปที่แอดเดรสของตัวแปรที่เป็นรูปแบบ char

# สตรัคเจอร์ (Structure)

# สตรัคเจอร์ (Structure)

คือ ข้อมูลแบบโครงสร้างที่นำเอาข้อมูลที่มีชนิดข้อมูลต่างกันมารวบรวมเข้าด้วยกัน แต่มีความสัมพันธ์ของข้อมูลแบบต่อกัน มาเก็บไว้ในโครงสร้างเดียวกัน

**\*\*เปรียบเทียบเสมือนกับสร้างชนิดข้อมูลขึ้นมาใช้งานเอง\*\***






# การสร้างสตรัคเจอร์

```
struct ชื่อสตรัคเจอร์ {  
    ชนิดข้อมูลตัวที่ 1 ตัวแปรที่ 1 ;  
    ชนิดข้อมูลตัวที่ 2 ตัวแปรที่ 2 ;  
    ....  
}
```



# การสร้างสตรัคเจอร์

```
struct user{  
    char name[20];  
    char gender;  
    int age;  
};
```



```
struct user emp1;  
strcpy(emp1.name,"kong");  
emp1.gender='M';  
emp1.age = 30;
```

# ช่องทางการสนับสนุน



ช่องยูทูป : <https://www.youtube.com/@KongRuksiamTutorial>



คอร์สเรียน : <https://www.udemy.com/user/kong-ruksiam/>



ซื้อของผ่าน Shopee : <https://shope.ee/3plB9kVnPd>



แฟนเพจ : <https://www.facebook.com/KongRuksiamTutorial/>