เขียนโปรแกรมภาษา C#

สำหรับผู้เริ่มต้น 2021

https://www.youtube.com/c/KongRuksiamOfficial/

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

เหมาะสำหรับ

- ผู้ที่สนใจเรียนรู้การเขียนโปรแกรมภาษา C# ด้วยตนเอง
- ปูพื้นฐานการเขียน Script C# ใช้งานในโปรแกรม Unity
- ไม่มีความรู้เรื่องการเขียนโปรแกรมก็เรียนได้
- เนื้อหาทั้งหมดเรียนฟรี!!

Introduction

ในปัจจุบันการเขียนโปรแกรมตระกูล .NET ไม่ได้มีแค่การพัฒนาโปรแกรมบน Desktop เท่านั้น แต่รวมไปถึงการพัฒนาโปรแกรมบนเว็บและในมือถือด้วย ซึ่งใน Visual Studio ก็รองรับการเขียนโปรแกรมทั้ง 3 รูปแบบนี้ด้วย ได้แก่

- พัฒนาโปรแกรมบน Desktop โดยใช้ส่วนของ Window Form App
- พัฒนาโปรแกรมบนเว็บ เช่น ASP.NET Core MVC เป็นต้น
- พัฒนาโปรแกรมบนสมาร์ทโฟน เช่น Xamarin เป็นต้น

ขอบเขตของเนื้อหา

- โครงสร้างและไวยากรณ์ภาษารวมถึงทฤษฎีการเขียนโปรแกรมด้วยภาษา C#
- แก้โจทย์ปัญหาเพื่อนำความรู้ไปใช้งานจริงได้

เครื่องมือที่ใช้

- Visual Studio Community (Free for Student)

โครงสร้างคำสั่งภาษา C#

```
using System; //เรียกใช้ Namespace
namespace BasicProgramming { // Namespace
        class Program{ //ชื่อคลาส
                static void Main(string[] args){ // เมธอด main
                        //คำสั่งต่างๆ
                }
        }
}
```


https://www.youtube.com/c/KongRuksiamOfficial/


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

องค์ประกอบพื้นฐาน

- **ขอบเขต (Block)** ใช้สัญลักษณ์ {} เพื่อบอกขอบเขตการทำงานของกลุ่มคำสั่งว่ามีจุดเริ่มต้นและสิ้นสุดที่ตำแหน่งใด
- **เครื่องหมายสิ้นสุดคำสั่ง** ใช้สัญลักษณ์ ;
- **คำอธิบาย (Comment)** ใช้สัญลักษณ์ // หรือ /* */

การแสดงผลทางจอภาพ

คำสั่งสำหรับแสดงข้อมูลออกทางจอภาพประกอบ
ด้วย 2 คำสั่งได้แก่ Write , WriteLine มีโครงสร้างดังนี้ คือ

Write(ข้อความ/ตัวเลข/ตัวแปร/เมธอด/ตัวดำเนินการ)

การแสดงผลทางจอภาพ

คำสั่งสำหรับแสดงข้อมูลออกทางจอภาพประกอบด้วย 2 คำสั่งได้แก่ Write , WriteLine มีโครงสร้างดังนี้ คือ

WriteLine(ข้อความ/ตัวเลข/ตัวแปร/เมธอด/ตัวดำเนินการ)

ตัวอย่างเช่น

```
Console.Write('A')

Console.Write('KongRuksiam')

Console.Write(100)

Console.Write(100+100)
```

การใช้ปีกกาบอกลำดับการแสดงผล

```
Console.Write(“ชื่อของฉัน คือ {0}”,”Kong”)
Console.Write(“ชื่อของฉัน คือ {0} อายุ {1}”,”ก้อง”,25)
```

การเขียนคำอธิบาย (Comment)

วิธีที่ 1 โดยใช้เครื่องหมาย Slash (/) ใช้ในการอธิบาย**คำสั่งสั้นๆในรูปแบบบรรทัดเดียว**

วิธีที่ 2 เขียนคำอธิบายไว้ในเครื่องหมาย /* ... */ ใช้ในการอธิบายคำสั่ง**ยาวๆหรือแบบหลายบรรทัด**

ตัวแปรและชนิดข้อมูล

ตัวแปร คือ ชื่อที่ถูกนิยามขึ้นมาเพื่อใช้เก็บค่าข้อมูลสำหรับนำไปใช้งานในโปรแกรมโดยข้อมูลประกอบด้วย ข้อความ ตัวเลข ตัวอักษร หรือผลลัพธ์จากการประมวลผลข้อมูล

Data Type	คำอธิบาย	Class และขนาด Bit
boolean	ค่าทางตรรกศาสตร์	8 (เก็บค่า True /False)
byte	ตัวเลขที่ไม่มีจุดทศนิยม	SByte(8)
short	ตัวเลขที่ไม่มีจุดทศนิยม	Int16
int	ตัวเลขที่ไม่มีจุดทศนิยม	Int32
long	ตัวเลขที่ไม่มีจุดทศนิยม	Int64
float	ตัวเลขที่มีจุดทศนิยม	Single(32)
double	ตัวเลขที่มีจุดทศนิยม	Double (64)
char	ตัวอักษร	Char(16)

ชนิดข้อมูลจะเป็นตัวกำหนดค่าที่สามารถเก็บได้ในตัวแปร
ยิ่งจำนวนของ bit มากเท่าไร แสดงว่าเราสามารถเก็บค่าได้มากเท่านั้น

Data Type	ค่าต่ำสุด	ค่าสูงสุด
boolean	ค่าทางตรรกศาสตร์	8 (เก็บค่า True /False)
byte	-128	127
short	-32768	32767
int	-2147483648	2147483647
long	-9223372036854775808	9223372036854775807
float	1.4E-45	3.4028235E38
double	4.9E-324	1.79769313486231 57E308
char	-	-

ชนิดข้อมูลจะเป็นตัวกำหนดค่าที่สามารถเก็บได้ในตัวแปร
ยิ่งจำนวนของ bit มากเท่าไร แสดงว่าเราสามารถเก็บค่าได้มากเท่านั้น

การนิยามตัวแปร

ชนิดข้อมูล ชื่อตัวแปร ;

ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น;

ให้นำค่าทางขวามือของเครื่องหมาย = ไปเก็บไว้ในตัวแปรที่อยู่ด้านซ้ายมือ

จำนวนเต็มและจำนวนทศนิยม

```
int x = 10;
หรือ
int x;
x=10;
```

```
double y = 10.24;
หรือ
double y;
y=10.24
```

```
float a=3.99f
หรือ
float a;
a=3.99f
```

ค่าคงที่ (Constant)

มีลักษณะการการใช้งานคล้ายกับตัวแปร แต่ค่าคงที่คือค่าจะไม่สามารถเปลี่ยนแปลงได้ ตอนประกาศใช้งานค่าคงที่ต้องมีการประกาศค่าเริ่มต้นเสมอ

การนิยามค่าคงที่ (Constant)

const ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น;

การนิยามค่าคงที่มักจะนิยามเป็นตัวพิมพ์ใหญ่

```
const float  PI = 3.14 ;
const int SIZE = 10;
```

กฎการตั้งชื่อ

- ขึ้นต้นด้วยตัวอักษร A-Z หรือ a-z หรือ @ หรือ _ เครื่องหมายขีดเส้นใต้ เท่านั้น
- อักษรตัวแรกห้ามเป็นตัวเลข
- ความยาวการตั้งชื่อไม่เกิน 63 ตัวอักษร
- Case Sensitive ตัวพิมพ์เล็ก-พิมพ์ใหญ่ มีความหมายต่างกัน
- ห้ามใช้อักขระพิเศษมาประกอบเป็นชื่อตัวแปร เช่น {} , % ^ และช่องว่าง เป็นต้น
- ไม่ซ้ำกับคำสงวนในภาษา C#

C# Keyword			
abstract	event	new	struct
as	explicit	null	switch
base	extern	object	this
bool	false	operator	throw
break	finally	out	true
byte	fixed	override	try
case	float	params	typeof
catch	for	private	uint
char	foreach	protected	ulong
checked	goto	public	unchecked
class	if	readonly	unsafe
const	implicit	ref	ushort
continue	in	return	using
decimal	int	sbyte	virtual
default	interface	sealed	volatile
delegate	internal	short	void
do	is	sizeof	while
double	lock	stackalloc	else
long	static	enum	namespace
string			

Format String สำหรับแสดงผลตัวเลข

รูปแบบอักขระ	ความหมาย
E หรือ e	Exponential (รูปแบบตัวเลขชี้กำลัง)
F หรือ f	Floating Point (เลขทศนิยม)
G หรือ g	แสดงตัวเลขรูปแบบสั้นที่สุด
N หรือ n	Number ใส่ comma คั่น ทุกๆ 3 หลัก
P หรือ p	Percentage (เลขเป็น %)
X หรือ x	Hexadecimal (เลขฐาน 16)

รับค่าผ่านทางคีย์บอร์ด

- Read() - อ่านค่าตัวอักษรผ่านทางคีย์บอร์ด (ASCII)
- ReadLine() - อ่านค่าผ่านทางคีย์บอร์ดใช้อ่าน String หรือ Double Float โดยส่งค่ากลับมาจะเป็น String

ตัวดำเนินการ (Operator)

กลุ่มของเครื่องหมายหรือสัญลักษณ์ที่ใช้ในการเขียนโปรแกรม

A+B

1. ตัวดำเนินการ (Operator)
2. ตัวถูกดำเนินการ (Operand)

ตัวดำเนินการทางคณิตศาสตร์

Operator	คำอธิบาย
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หารเอาเศษ

ฟังก์ชั่นทางคณิตศาสตร์

ชื่อฟังก์ชั่น	การทำงาน
Math.Abs(x)	ค่าสัมบูรณ์ของ x (|x|)
Math.Ceiling(x)	ปัดเศษทศนิยมขึ้นทุกกรณี
Math.Floor(x)	ปัดเศษทศนิยมทิ้งทุกกรณี
Math.Round(x)	ปัดเศษทศนิยม >=.5 เป็นต้นไปปัดเศษขึ้น ถ้า <.5 ให้ปัดเศษลง
Math.Pow(x,y)	x ยกกำลัง y
Math.Sqrt(x)	รากที่สองของ x
Math.PI	ค่าคงที่ π มีค่าประมาณ 3.141592653
Math.E	ค่าคงที่ e มีค่าประมาณ 2.718281828

ตัวดำเนินการเปรียบเทียบ

**** ชนิดข้อมูล boolean

Operator	คำอธิบาย
==	เท่ากับ
!=	ไม่เท่ากับ
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าเท่ากับ
<=	น้อยกว่าเท่ากับ

ตัวดำเนินการเพิ่มค่า - ลดค่า

Operator	รูปแบบการเขียน	ความหมาย
++ (Prefix)	++a	เพิ่มค่าให้ a ก่อน 1 ค่าแล้วนำไปใช้
++ (Postfix)	a++	นำค่าปัจจุบันใน a ไปใช้ก่อนแล้วค่อยเพิ่มค่า
-- (Prefix)	--b	ลดค่าให้ b ก่อน 1 ค่าแล้วนำไปใช้
-- (Postfix)	b--	นำค่าปัจจุบันใน b ไปใช้ก่อนแล้วค่อยลดค่า

Compound Assignment

Assignment	รูปแบบการเขียน	ความหมาย
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x/=y	x=x/y
%=	x%=y	x=x%y

ลำดับความสำคัญของตัวดำเนินการ

ลำดับที่	เครื่องหมาย	ลำดับการทำงาน
1	( )	
2	++ , --	ซ้ายไปขวา
3	* , / , %	ซ้ายไปขวา
4	+ , -	ซ้ายไปขวา
5	< , <= , > , >=	ซ้ายไปขวา
6	== , !=	ซ้ายไปขวา
7	&& (AND)	ซ้ายไปขวา
8	|| (OR)	ซ้ายไปขวา
9	= , += , -= , *= , /= , %=	ขวาไปซ้าย

กรณีศึกษา

1. 5+8 *9
2. 10 - 4+2
3. 10 - (2+1)
4. 5 * 2 – 40 / 5
5. 7+8/2+25

การแปลงชนิดข้อมูล (Type Casting)

1.Widening Casting
คือการแปลงข้อมูลที่มีขนาดเล็กไปสู่ข้อมูลขนาดใหญ่ (แบบอัตโนมัติ)

byte -> short -> int -> long -> float -> double

2.Narrowing Casting
คือการแปลงข้อมูลที่มีขนาดใหญ่ไปสู่ข้อมูลที่มีขนาดเล็ก (ทำเอง)

double -> float -> long -> int -> char -> short -> byte

การแปลงชนิดข้อมูล (Type Casting)

ข้อมูลทุกชนิดสามารถ
แปลงเป็น String ได้ทั้งหมด

การแปลงชนิดข้อมูล

จะใช้ในการแปลงชนิดข้อมูล เช่น

1. แปลงจากชนิดข้อมูลหนึ่งไปเป็นอีกชนิดหนึ่งได้
2. แปลงจากตัวเลขเป็นสตริง
3. แปลงจากสตริงเป็นตัวเลข

การแปลงชนิดข้อมูล

โดยใช้งาน Class และ Method ที่อยู่ใน C# มีให้บริการอยู่ 4 รูปแบบด้วยกัน คือ

1. ใช้ Class Convert
2. ใช้ TryParse
3. ใช้ Parse (แปลงข้อความเป็นตัวเลข)
4. toString() - ทำให้เป็น String ทั้งหมด

Data Type	คำอธิบาย	Class และขนาด Bit
boolean	ค่าทางตรรกศาสตร์	8 (เก็บค่า True /False)
byte	ตัวเลขที่ไม่มีจุดทศนิยม	SByte(8)
short	ตัวเลขที่ไม่มีจุดทศนิยม	Int16
int	ตัวเลขที่ไม่มีจุดทศนิยม	Int32
long	ตัวเลขที่ไม่มีจุดทศนิยม	Int64
float	ตัวเลขที่มีจุดทศนิยม	Single(32)
double	ตัวเลขที่มีจุดทศนิยม	Double (64)
char	ตัวอักษร	Char(16)

ชนิดข้อมูลจะเป็นตัวกำหนดค่าที่สามารถเก็บได้ในตัวแปร
ยิ่งจำนวนของ bit มากเท่าไร แสดงว่าเราสามารถเก็บค่าได้มากเท่านั้น

Data Type	ค่าต่ำสุด	ค่าสูงสุด
boolean	ค่าทางตรรกศาสตร์	8 (เก็บค่า True /False)
byte	-128	127
short	-32768	32767
int	-2147483648	2147483647
long	-9223372036854775808	9223372036854775807
float	1.4E-45	3.4028235E38
double	4.9E-324	1.79769313486231 57E308
char	-	-

ชนิดข้อมูลจะเป็นตัวกำหนดค่าที่สามารถเก็บได้ในตัวแปร
ยิ่งจำนวนของ bit มากเท่าไร แสดงว่าเราสามารถเก็บค่าได้มากเท่านั้น

การใช้งาน Convert Class

byte x = Convert.ToByte(ตัวเลขหรือสตริงที่ต้องการแปลงเป็น byte)
short y = Convert.ToInt16(ตัวเลขหรือสตริงที่ต้องการแปลงเป็น short)
int z = Convert.ToInt32(ตัวเลขหรือสตริงที่ต้องการแปลงเป็น int)
long a = Convert.ToInt64(ตัวเลขหรือสตริงที่ต้องการแปลงเป็น long)
float b = Convert.ToSingle(ตัวเลขหรือสตริงที่ต้องการแปลงเป็น float)
double d = Convert.ToDouble(ตัวเลขหรือสตริงที่ต้องการแปลงเป็น double)
string d = Convert.ToString(ตัวเลขที่ต้องการแปลงเป็น String)

การใช้งาน Parse

Parse(“ข้อความ”)

```
int number1 = Int.parse(“200”);
double number2 = double.parse(“4.00”)
double number3 =
double.parse(textbox.text)
```

การใช้งาน TryParse

TryParse(“ข้อความ”,out result)

เก็บสถานะการแปลงลงใน success (True/False)
เก็บผลลัพธ์ลงใน result

```
success= Int.TryParse(“200”,out result);
success= Int.TryParse(“200”,out result);
```

ความแตกต่าง

- Parse(“ข้อความ”) - เป็นการแปลงข้อมูลชนิด string หรือข้อความให้เป็นข้อมูลชนิดที่ต้องการ
- TryParse มีการจัดการ Exception (ข้อผิดพลาด) แล้วส่งค่ากลับมาเป็น True / False
- toString() แปลงอะไรก็ได้เป็น String

Assignment 1: โปรแกรมคำนวณค่าดัชนีมวลกาย (BMI)

$$\text{ดัชนีมวลกาย (BMI)} = \frac{\text{น้ำหนักตัว (กิโลกรัม)}}{\text{ส่วนสูง (เมตร)}^2}$$

ยกตัวอย่าง เช่น ถ้ามีน้ำหนัก 60 กิโลกรัม และสูง 1.55 ม.
ดัชนีมวลกาย (BMI) = 24.97

โครงสร้างควบคุม (Control Structure)

คือ กลุ่มคำสั่งที่ใช้ควบคุมการทำงานของโปรแกรม

- แบบลำดับ (Sequence)
- แบบมีเงื่อนไข (Condition)
- แบบทำซ้ำ (Loop)

แบบมีเงื่อนไข (Condition)

กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่างๆ ภายในโปรแกรมมาทำงาน

- if
- Switch..Case

รูปแบบคำสั่งแบบเงื่อนไขเดียว

- **if statement**

เป็นคำสั่งที่ใช้กำหนดเงื่อนไขในการตัดสินใจทำงานของโปรแกรม

ถ้าเงื่อนไขเป็นจริงจะทำตามคำสั่งต่างๆ ที่กำหนดภายใต้เงื่อนไขนั้นๆ

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}
```

รูปแบบคำสั่งแบบ 2 เงื่อนไข

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}else{
    คำสั่งเมื่อเงื่อนไขเป็นเท็จ ;
}
```

รูปแบบคำสั่งแบบหลายเงื่อนไข

```
if(เงื่อนไขที่ 1){
        คำสั่งเมื่อเงื่อนไขที่ 1 เป็นจริง ;
}else if(เงื่อนไขที่ 2){
        คำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;
}else if(เงื่อนไขที่ 3){
        คำสั่งเมื่อเงื่อนไขที่ 3 เป็นจริง ;
}else{
        คำสั่งเมื่อทุกเงื่อนไขเป็นเท็จ ;
}
```

โจทย์ปัญหา : โปรแกรมตัดเกรดอย่างง่าย

คำนวณเกรดจากคะแนนสอบของนักเรียน โดยมีเกณฑ์ดังนี้ คือ

- คะแนน 80 ขึ้นไป ได้เกรด A
- คะแนน 70 ขึ้นไป ได้เกรด B
- คะแนน 60 ขึ้นไป ได้เกรด C
- คะแนน 50 ขึ้นไป ได้เกรด D
- น้อยกว่า 50 คะแนน ได้เกรด F

ข้อควรระวังการเขียน if เพื่อตรวจสอบเงื่อนไข

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}
```

ตัวดำเนินการทางตรรกศาสตร์

Operator	คำอธิบาย
&&	AND
||	OR
!	NOT

ตัวดำเนินการทางตรรกศาสตร์

a	!a	a	b	a && b	a || b
true	false	false	false	false	false
false	true	false	true	false	true
		true	false	false	true
		true	true	true	true

โจทย์ปัญหา : เกณฑ์การคัดเลือกบุคลากร

AND (และ) , &&

ผู้สมัครเป็นเพศชาย **และ** มีส่วนสูง 160 ซม. เป็นต้นไป **(ผ่านเกณฑ์)**

OR (หรือ) , ||

ผู้สมัครเป็นเพศชาย **หรือ** มีส่วนสูง 160 ซม. เป็นต้นไป **(ผ่านเกณฑ์)**

NOT (ไม่) , !

ผู้สมัคร**ไม่ได้**เป็นเพศชาย

โจทย์ปัญหา : โปรแกรมตัดเกรดอย่างง่าย

คำนวณเกรดจากคะแนนสอบของนักเรียน คะแนนเต็ม 100 คะแนน โดยมีเกณฑ์ดังนี้ คือ

- คะแนน 80 - 100ได้เกรด A
- คะแนน 70 - 79 ได้เกรด B
- คะแนน 60 - 69 ได้เกรด C
- คะแนน 50 - 59ได้เกรด D
- น้อยกว่า 50 คะแนน ได้เกรด F
- ป้อนค่าอื่นแสดงข้อความว่า ไม่พบข้อมูล

if..else แบบลดรูป (Ternary Operator)

ตัวแปร = (เงื่อนไข) ? คำสั่งเมื่อเงื่อนไขเป็นจริง : คำสั่งเมื่อเงื่อนไขเป็นเท็จ;

```
if(เงื่อนไข){
    คำสั่งเมื่อเงื่อนไขเป็นจริง
}else{
    คำสั่งเมื่อเงื่อนไขเป็นเท็จ
}
```

การเขียน if ซ้อน if

```
if(เงื่อนไขที่ 1){
    if(เงื่อนไขที่ 2 ){
        คำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;
    }
}
```

โจทย์ปัญหา : โปรแกรมคำนวณส่วนลด

ถ้าชำระเงินเกิน 15,000 บาทจะได้รับส่วนลดแต่ถ้าน้อยกว่าจะไม่ได้รับส่วนลด

ชำระเงิน 15,000 บาท ส่วนลด 10%

ชำระเงิน 20,000 บาท ส่วนลด 20%

ชำระเงิน 30,000 บาท ส่วนลด 30%

แบบมีเงื่อนไข (Condition)

กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่างๆ ภายในโปรแกรมมาทำงาน

- **Switch..Case**

Switch เป็นคำสั่งที่ใช้กำหนดเงื่อนไขคล้ายๆกับ if แต่จะเลือกเพียงหนึ่งทางเลือกออกมาทำงานโดยนำค่าในตัวแปรมากำหนดเป็นทางเลือกผ่านคำสั่ง case

รูปแบบคำสั่ง

```
switch(สิ่งที่ต้องการตรวจสอบ) {

      case ค่าที่ 1 : คำสั่งที่ 1;
            break;
       case ค่าที่ 2 : คำสั่งที่ 2;
            break;
           ......
       case ค่าที่ N : คำสั่งที่ N;
            break;

      default : คำสั่งเมื่อไม่มีค่าที่ตรงกับที่ระบุใน case
}
```

***คำสั่ง break จะทำให้โปรแกรมกระโดดออกไปทำงานนอกคำสั่ง switch ถ้าไม่มีคำสั่ง break โปรแกรมจะทำคำสั่งต่อไปเรื่อยๆ จนจบการทำงาน

โจทย์ปัญหา : โปรแกรมคำนวณเลข

ให้ป้อนตัวเลข 2 จำนวน แล้วเลือกรูปแบบการคำนวณตัวเลขผ่านตัวเลือกที่กำหนด

- ถ้าพิมพ์เลข 1 เป็นการบวกเลข
- ถ้าพิมพ์เลข 2 เป็นการลบเลข
- ถ้าพิมพ์ตัวเลขอื่น แจ้งว่าข้อมูลไม่ถูกต้อง

แบบทำซ้ำ (Loop)

กลุ่มคำสั่งที่ใช้ในการวนรอบ (loop) โปรแกรมจะทำงานไปเรื่อยๆจนกว่าเงื่อนไขที่กำหนดไว้จะเป็นเท็จ จึงจะหยุดทำงาน

- While
- For
- Do..While

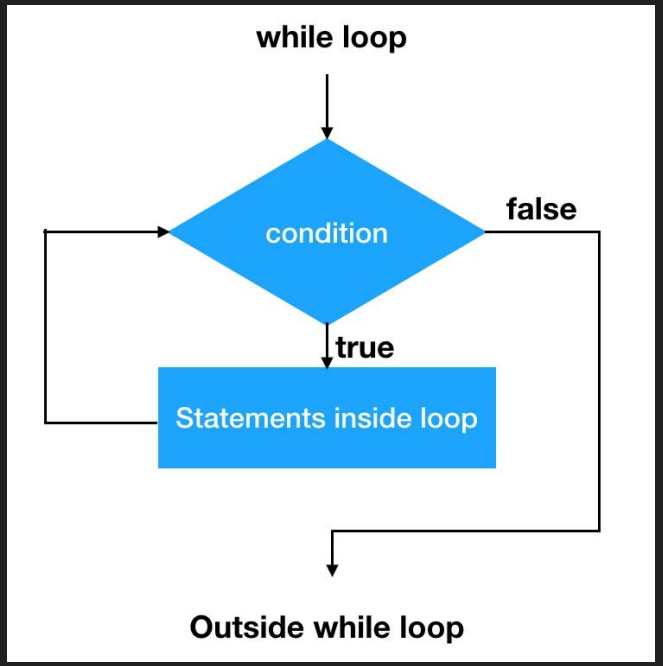
คำสั่ง While

- **While Loop**

จะทำงานตามคำสั่งภายใน while ไปเรื่อยๆเมื่อเงื่อนไขที่กำหนดเป็นจริง

```
while(เงื่อนไข){
    คำสั่งที่จะทำซ้ำเมื่อเงื่อนไขเป็นจริง ;
}
```

คำสั่ง While



1.เช็คเงื่อนไขถ้าเป็นจริงให้ทำคำสั่งซ้ำใน Statement

2.ถ้าเป็นเท็จให้ออกจาก Loop

https://cdn.journaldev.com/wp-content/uploads/2017/10/while-loop-java.png.webp

คำสั่ง For

- **For Loop**

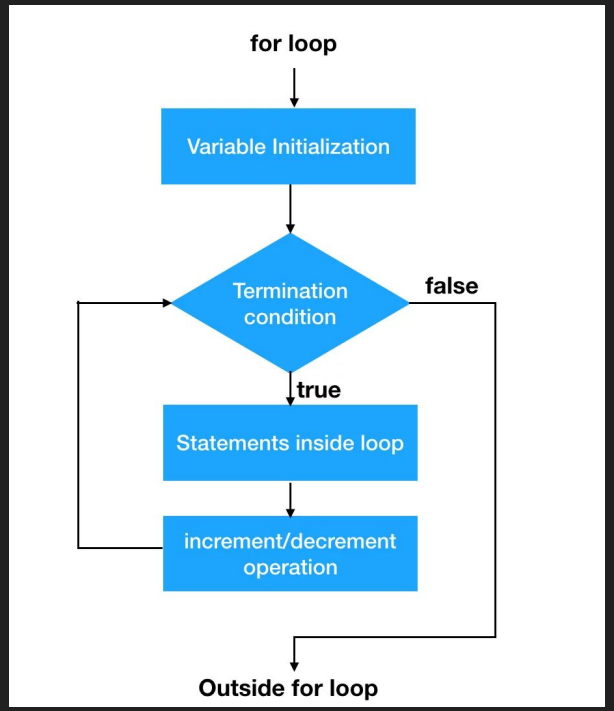
เป็นรูปแบบการซ้ำที่ใช้ในการตรวจสอบเงื่อนไขการทำงาน มีการกำหนดค่าเริ่มต้นและเปลี่ยนค่าไปพร้อมๆกัน เมื่อเงื่อนไขในคำสั่ง for เป็นจริง ก็จะทำงานตามคำสั่งที่แสดงไว้ภายในคำสั่ง for ไปเรื่อยๆ

โครงสร้างคำสั่ง

```
for(ค่าเริ่มต้นของตัวแปร; เงื่อนไข; เปลี่ยนแปลงค่าตัวแปร) {
        คำสั่งเมื่อเงื่อนไขเป็นจริง;
}
```

```
for(int i = 1;i<=10;i++) {
    คำสั่งเมื่อเงื่อนไขเป็นจริง;
}
```

คำสั่ง For



1.กำหนดค่าเริ่มต้น

2.เช็คเงื่อนไขถ้าเป็นจริงให้ทำคำสั่งซ้ำใน Statement

3.ถ้าเป็นเท็จให้ออกจาก Loop

https://cdn.journaldev.com/wp-content/uploads/2017/10/java-for-loop.png.webp

คำสั่ง Do..While

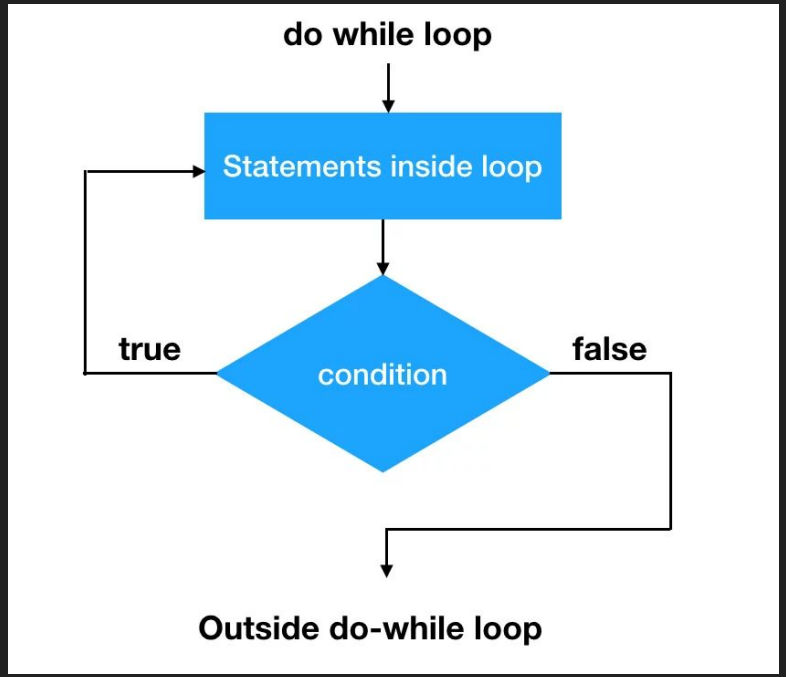
- **Do..While**

โปรแกรมจะทำงานตามคำสั่งอย่างน้อย 1 รอบ เมื่อทำงานเสร็จจะมาตรวจสอบเงื่อนไขที่คำสั่ง while ถ้าเงื่อนไขเป็นจริงจะวนกลับขึ้นไปทำงานที่คำสั่งใหม่อีกรอบ แต่ถ้าเป็นเท็จจะหลุดออกจากลูป

โครงสร้างคำสั่ง

```
do {
    คำสั่งต่างๆ เมื่อเงื่อนไขเป็นจริง;
 } while(เงื่อนไข);
```

คำสั่ง Do..While



1.ทำงานคำสั่งใน Statement

2.เช็คเงื่อนไขถ้าเป็นจริงให้กลับไปทำซ้ำใน Statement

3.ถ้าเป็นเท็จให้ออกจาก Loop

https://cdn.journaldev.com/wp-content/uploads/2017/10/java-do-while-loop-1.png.webp

คำสั่งที่เกี่ยวข้องกับ Loop

- **break** ถ้าโปรแกรมพบคำสั่งนี้จะหลุดจากการทำงานในลูปทันที เพื่อไปทำคำสั่งอื่นที่อยู่นอกลูป

- **continue** คำสั่งนี้จะทำให้หยุดการทำงานแล้วย้อนกลับไปเริ่มต้นการทำงานที่ต้นลูปใหม่

ข้อแตกต่างและการใช้งาน Loop

- For ใช้ในกรณีรู้จำนวนรอบที่ชัดเจน
- While ใช้ในกรณีที่ไม่รู้จำนวนรอบ
- Do..while ใช้ในกรณีที่อยากให้ลองทำก่อน 1 รอบ แล้วทำซ้ำไปเรื่อยๆตราบเท่าที่เงื่อนไขเป็นจริง

โจทย์ปัญหา : หาผลรวมและค่าเฉลี่ย

- จำนวนตัวเลข : 5 จำนวน (n=5)
- รับค่าตัวเลขผ่านแป้นพิมพ์
- แสดงผลรวมตัวเลข 5 จำนวน (summation)
- แสดงค่าเฉลี่ยตัวเลขจากสมการ

average = summation / n

โจทย์ปัญหา : หาผลรวมและค่าเฉลี่ย (ไม่จำกัด)

- จำนวนตัวเลข : ไม่จำกัด (n)
- รับค่าตัวเลขผ่านแป้นพิมพ์
- ถ้าป้อนเลขติดลบให้จบการทำงาน
- แสดงผลรวมตัวเลขตามจำนวนที่ป้อน (summation)
- แสดงค่าเฉลี่ยตัวเลขจากสมการ

$$average = summation / n$$

โจทย์ปัญหา : หาตัวเลขต่ำสุด - สูงสุด

- จำนวนตัวเลข : ไม่จำกัด (n)
- รับค่าตัวเลขผ่านแป้นพิมพ์
- ถ้าป้อนเลขติดลบให้จบการทำงาน
- ตรวจสอบว่าเลขที่ป้อนมีค่าสูงสุดเท่าใด
- ตรวจสอบว่าเลขที่ป้อนมีค่าต่ำสุดเท่าใด

โจทย์ปัญหา : การจ่ายธนบัตรของ ATM

- รับตัวเลข 100 เป็นต้นไป
- ตู้ ATM จะจ่ายธนบัตรฉบับละ 100 , 500 , 1,000 บาทเท่านั้น
- ถ้าป้อนเลขหลักหน่วย - หลักสิบจะให้กลับไปป้อนจำนวนเงินอีกครั้ง

โจทย์ปัญหา : การจ่ายธนบัตรของ ATM

- ต้องการกดเงิน 1,800 บาท
- 1,800 หารด้วย 1000 จะได้ธนบัตร 1,000 จำนวน 1 ฉบับ
- หาเศษจากการหาร 1,800 ด้วย 1,000 จะได้เศษ 800
- นำ 800 หารด้วย 500 จะได้ธนบัตร 500 จำนวน 1 ฉบับ
- หาเศษจากการหาร 800 ด้วย 500 จะได้เศษ 300
- นำ 300 หารด้วย 100 จะได้ธนบัตร 100 จำนวน 3 ฉบับ

เขียนโปรแกรมภาษา C#

สำหรับผู้เริ่มต้น Phase 2

https://www.youtube.com/c/KongRuksiamOfficial/

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


Visual Studio

Array

ข้อจำกัดของชนิดข้อมูลพื้นฐาน

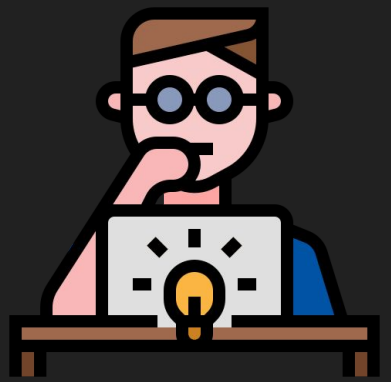
การประกาศตัวแปรแต่ละครั้ง

ตัวแปร 1 ตัวสามารถเก็บข้อมูลได้แค่ 1 ค่าเท่านั้น เช่น

```
int number = 1;
```

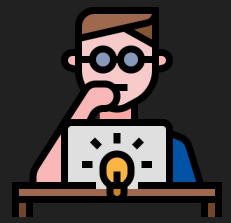
ถ้าอยากเก็บเลข 10 ค่าต้องทำอย่างไร ?

ต้องประกาศตัวแปร 10 ตัวแปร หรือไม่ ?



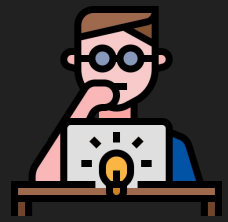
Array คืออะไร

ความหมายที่ 1 ชุดของตัวแปรที่อยู่ในรูปลำดับใช้เก็บค่าข้อมูลให้อยู่ในกลุ่มเดียวกัน ข้อมูลภายในอาร์เรย์จะถูกเก็บบนหน่วยความจำในตำแหน่งที่ต่อเนื่องกัน โดยขนาดของอาร์เรย์จะเล็ก หรือใหญ่ขึ้นกับจำนวนมิติที่กำหนดขึ้น



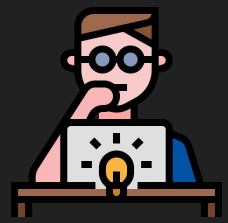
Array คืออะไร

ความหมายที่ 2 เป็นตัวแปรที่ใช้ในการเก็บข้อมูลที่มีลำดับที่ต่อเนื่อง ซึ่งข้อมูลมีค่าได้หลายค่าโดยใช้ชื่ออ้างอิงได้เพียงชื่อเดียว และใช้หมายเลขกำกับ **(index)** ให้กับตัวแปรเพื่อจำแนกความแตกต่างของค่าตัวแปรแต่ละตัว



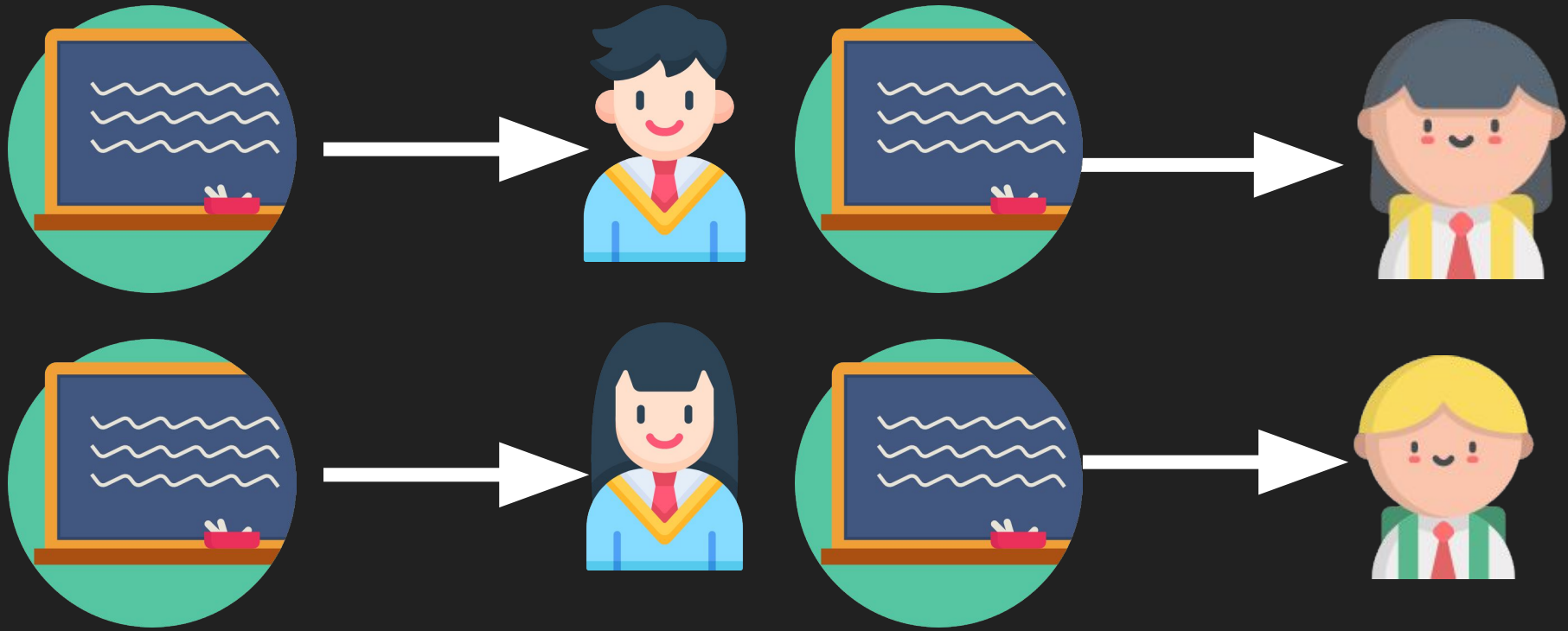
คุณสมบัติของ Array

1. ใช้เก็บกลุ่มของข้อมูล
2. ข้อมูลที่อยู่ในอาร์เรย์จะเรียกว่าสมาชิก หรือ อิลิเมนต์ (element)
3. แต่ละอิลิเมนต์ (element) จะเก็บค่าข้อมูล (value) และ อินเด็กซ์ (Index) เอาไว้
4. Index หมายถึงคีย์ของอาร์เรย์ใช้อ้างอิงตำแหน่งของ element **เริ่มต้นที่ 0**
5. สมาชิกใน array ต้องมี**ชนิดข้อมูลเหมือนกัน**
6. สมาชิกใน array จะถูกคั่นด้วยเครื่องหมาย comma



ตัวแปร = ห้องเรียน

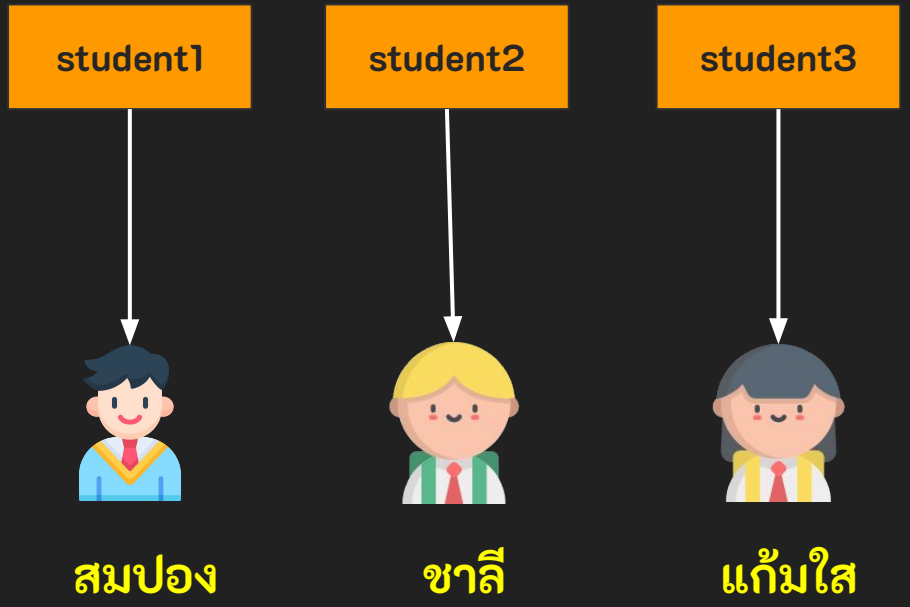
ค่าที่เก็บในตัวแปร = นักเรียน



การสร้างตัวแปรแบบปกติ

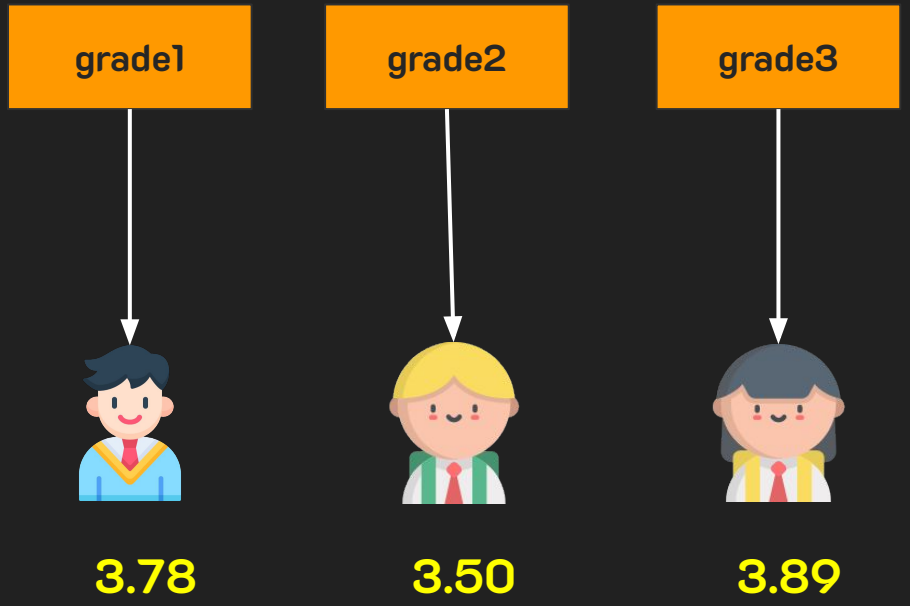
ตัวอย่างการประกาศตัวแปรแบบปกติ

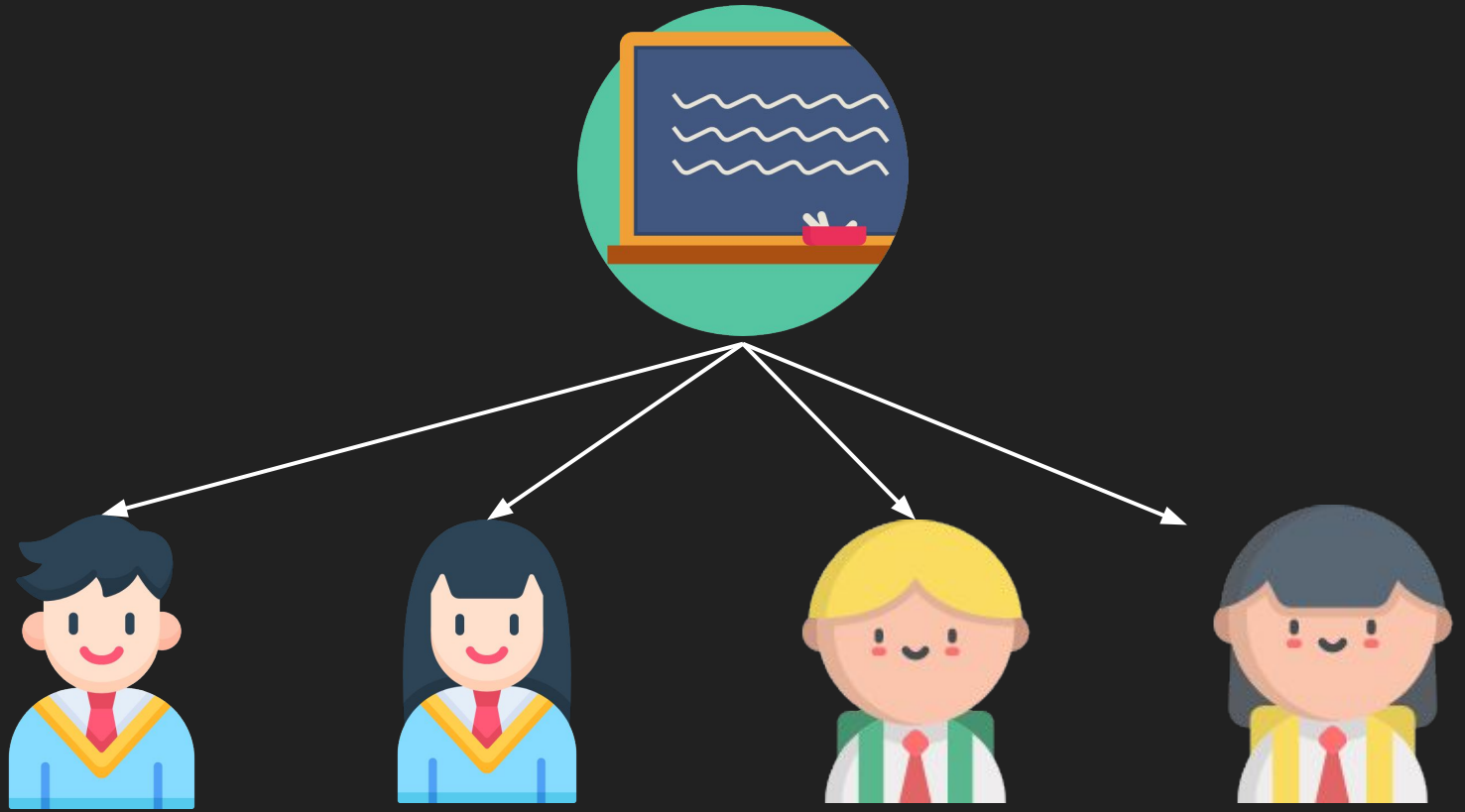
```
string student1 = “สมปอง”
string student2 = “ชาลี”
string student3 = “แก้มใส”
```



ตัวอย่างการประกาศตัวแปรแบบปกติ

```
double grade1 = 3.78
double grade1 = 3.50
double grade3 = 3.89
```

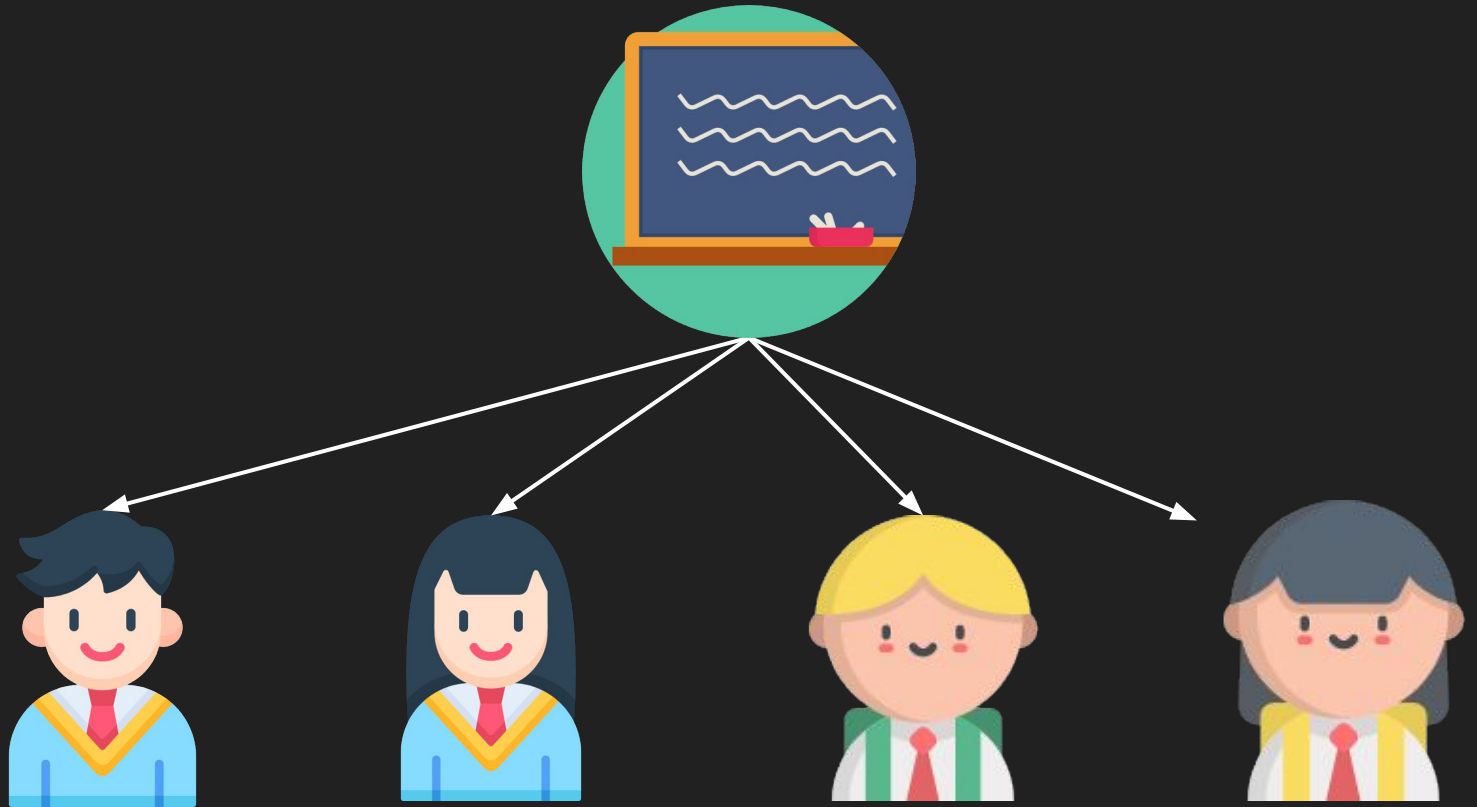




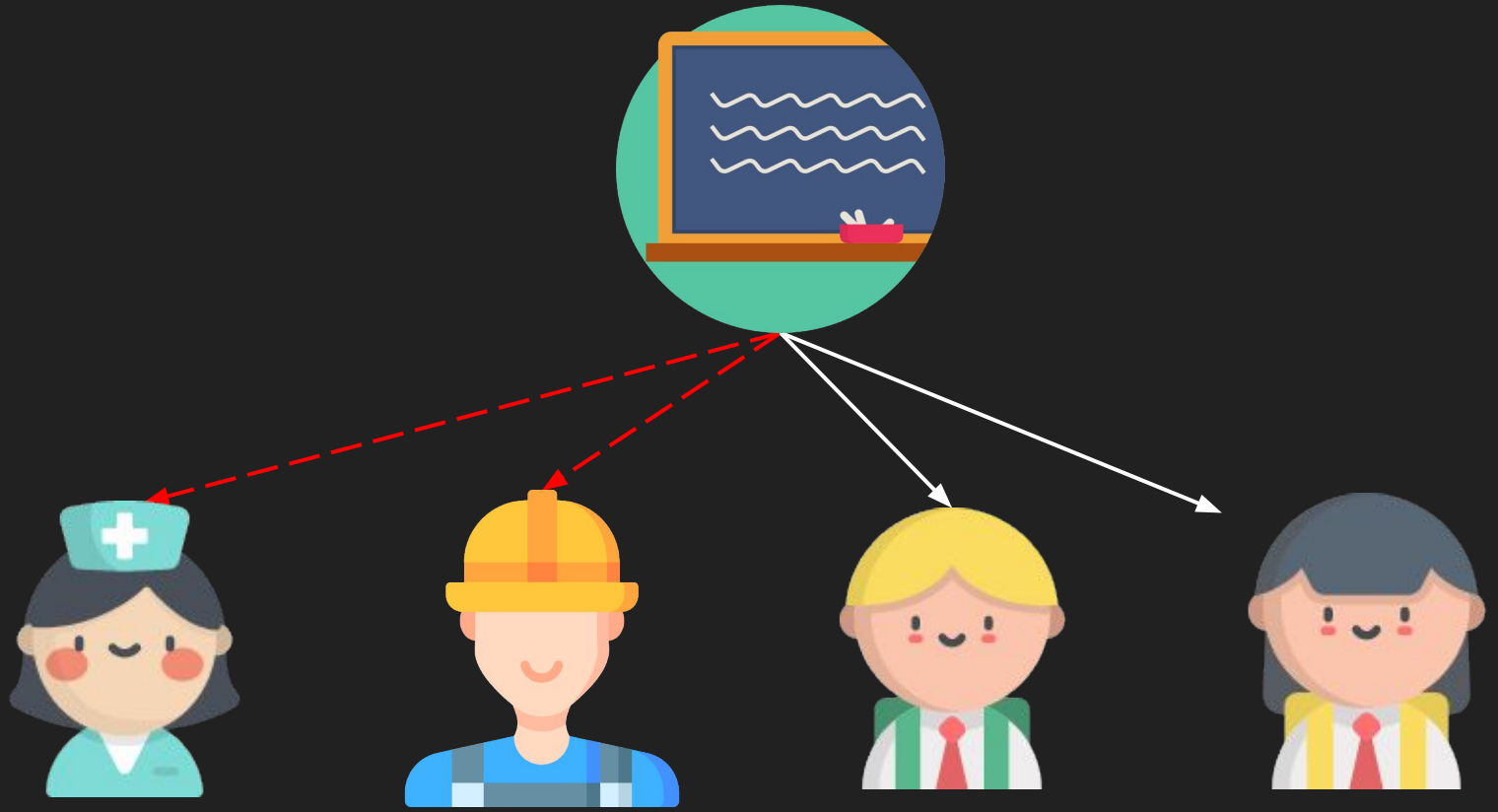
การสร้างตัวแปรแบบ Array

0 1 2 3

ใช้หมายเลขกำกับอ้างอิงข้อมูล (เป็นลำดับ)



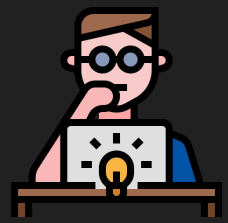
ข้อจำกัด คือ มีขนาดที่แน่นอน



ข้อจำกัด คือ ต้องเป็นนักเรียนเท่านั้น!!

สรุป **

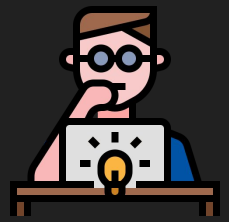
1. ใช้เก็บกลุ่มของข้อมูล **ที่มีชนิดข้อมูลเดียวกัน**
2. ใช้ตัวแปรชื่อเดียวกัน
3. ใช้หมายเลขกำกับเพื่ออ้างอิงตำแหน่งของข้อมูลใน Array



การสร้าง Array

มี 2 รูปแบบ คือ

1. แบบไม่ประกาศค่าเริ่มต้น ใช้สัญลักษณ์ []
2. แบบประกาศค่าเริ่มต้น ใช้สัญลักษณ์ {}



การสร้าง Array

1. **แบบไม่ประกาศค่าเริ่มต้น**

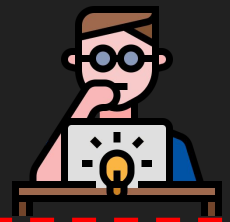
   ชนิดข้อมูล [] ชื่อตัวแปร = new ชนิดข้อมูล [ขนาด];

   เช่น int[] number = new int[4];

2. **แบบประกาศค่าเริ่มต้น**

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

   เช่น int[] number = {10,20,30,40};



การกำหนดค่าสมาชิกใน Array

```
number[0] = 10;
number[1] = 20;
number[2] = 30;
number[3] = 40;
```

การสร้าง Array แบบกำหนดค่าเริ่มต้น

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

10	20	30	40

```
string [] students = {“สมชาย”,”แก้วตา”};
```

สมชาย	แก้วตา

จำนวนสมาชิกใน Array (Length)

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

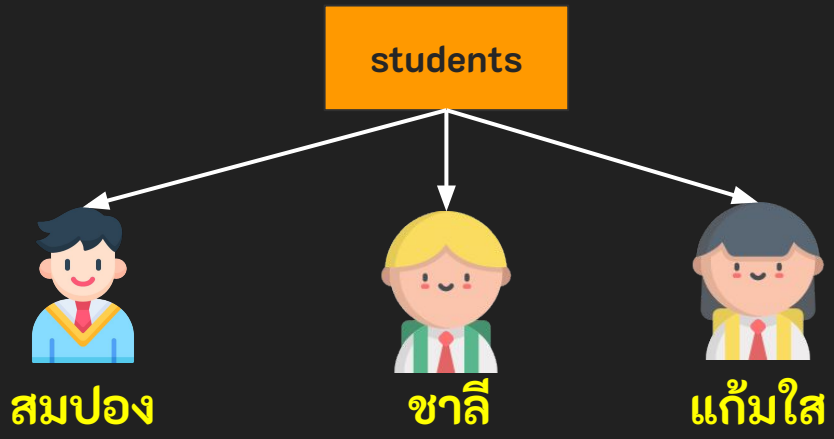
การเข้าถึงสมาชิก Array

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


```
string [] students = {“สมชาย”,”แก้วตา”};
```

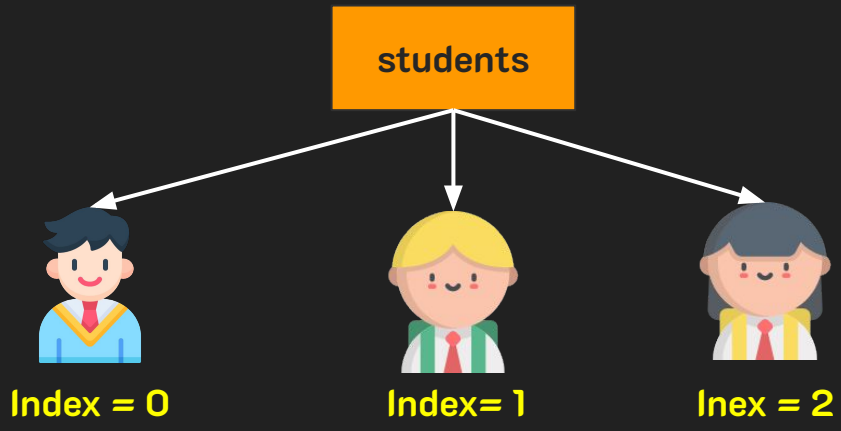
ตัวอย่างที่ 1

string [] students={“สมปอง”,”ชาลี”,”แก้มใส”}



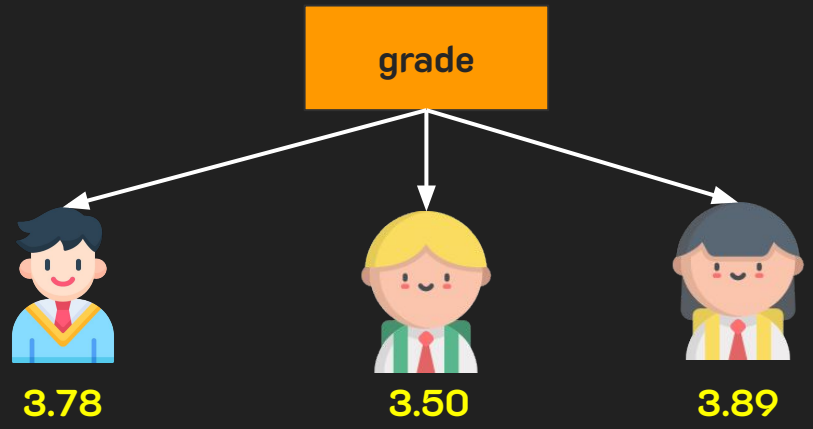
ตัวอย่างที่ 1

string [] students={“สมปอง”,”ชาลี”,”แก้มใส”}



ตัวอย่างที่ 2

```
double [] grade ={3.78,3.50,3.89};
```

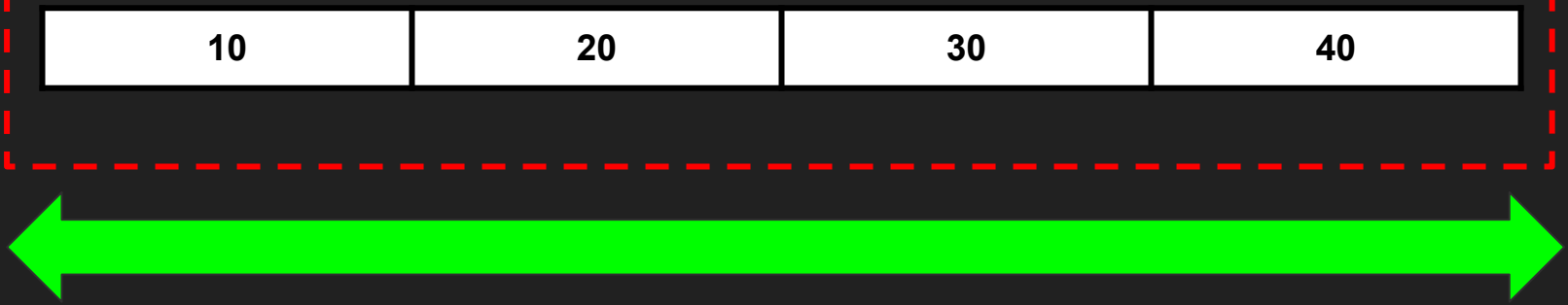


การเปลี่ยนแปลงข้อมูลสมาชิก Array

```
int[] number = {10, 20, 30, 40};
number[1] = 100;
string [] students = {“สมชาย”,”แก้วตา”};
students [1] = “แก้มใส”;
```

นับจำนวนสมาชิกใน Array(Length)

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



นับจำนวนสมาชิกใน Array(Length)

```
int[] number = {10, 20, 30, 40};
number.Length;
string [] pets = {“แมว”,”กระต่าย”};
pets.Length;
```

การเข้าถึงสมาชิกด้วย For Loop

```
string [] pets = {“แมว”,”กระต่าย”};

for (int i = 0; i < pets.length; i++) {
    // แสดงผล
}
```

การเข้าถึงสมาชิกด้วย ForEach

```
string [] pets = {“แมว”,”กระต่าย”};

  foreach (string name : pets) {
     // แสดงผล
  }
```

รูปแบบของ Array

- Array 1 มิติ (Single Dimensional Array)
- Array หลายมิติ (Multidimensional Array)
- Jagged Array

Array 2 มิติ

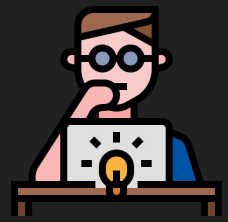
รูปแบบของ Array 1 มิติ

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

รูปแบบของ Array 2 มิติ

ความหมายของ Array 2 มิติ

- Array ที่มีข้อมูลสมาชิกภายในเป็น Array (Array ซ้อน Array) เปรียบเสมือนกับ Array ย่อยหรือ Array รูปแบบ matrix
- มีโครงสร้างเป็นรูปแบบแถว (แนวนอน) และคอลัมน์ (แนวตั้ง)
- มีจำนวนสมาชิกในแต่ละ Array ย่อย**จำนวนเท่ากัน**

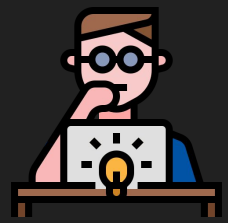


สร้าง Array 2 มิติ (แบบไม่กำหนดค่าเริ่มต้น)

ชนิดข้อมูล [,] ชื่อตัวแปร = new ชนิดข้อมูล [จำนวนแถว, จำนวนคอลัมน์]

ตัวอย่าง เช่น

- string [,] students = new string[3,4]
- int [,] numbers = new int [3,4]



สร้าง Array 2 มิติ (แบบกำหนดค่าเริ่มต้น)

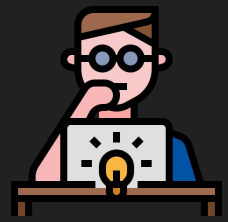
ชนิดข้อมูล [,] ชื่อตัวแปร = new ชนิดข้อมูล [แถว,คอลัมน์]{

{Array ตัวที่ 1},

{Array ตัวที่ 2},

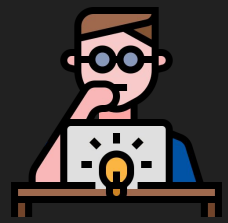
{Array ตัวที่ 3}

}



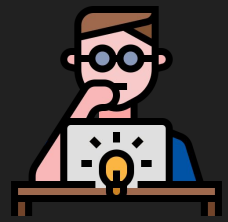
สร้าง Array 2 มิติ (แบบกำหนดค่าเริ่มต้น)

```
ชนิดข้อมูล [,] ชื่อตัวแปร = {
        {Array ตัวที่ 1},
        {Array ตัวที่ 2},
        {Array ตัวที่ 3}
}
```



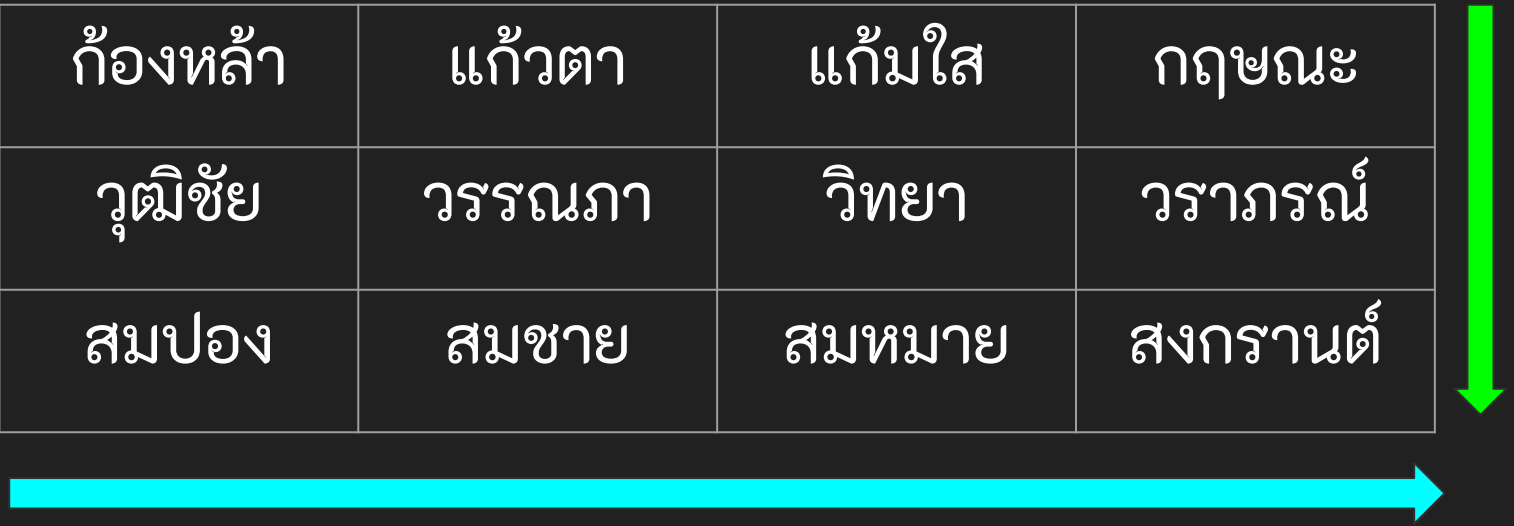
ตัวอย่างการสร้าง Array 2 มิติ

```
string [,] students = {
    {“ก้องหล้า”,”แก้วตา”,”แก้มใส”,”กฤษณะ”},
    {“วุฒิชัย”,”วรรณภา”,”วิทยา”,”วราภรณ์”},
    {“สมปอง”,”สมชาย”,”สมหมาย”,”สงกรานต์”}
}
```



โครงสร้างของ Array 2 มิติ

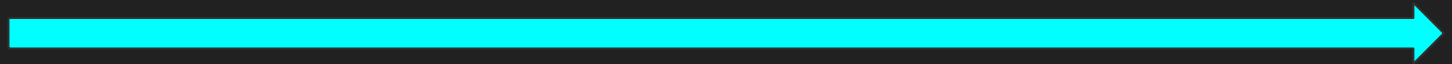
	คอลัมน์ที่ 0	คอลัมน์ที่ 1	คอลัมน์ที่ 2	คอลัมน์ที่ 3
แถวที่ 0	ก้องหล้า	แก้วตา	แก้มใส	กฤษณะ
แถวที่ 1	วุฒิชัย	วรรณภา	วิทยา	วราภรณ์
แถวที่ 2	สมปอง	สมชาย	สมหมาย	สงกรานต์



การเข้าถึงสมาชิกใน Array 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]

อ้างอิงเลขแถวและเลขคอลัมน์

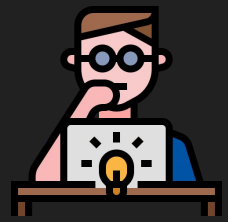


การเข้าถึงสมาชิกใน Array 2 มิติ

	คอลัมน์ที่ 0	คอลัมน์ที่ 1	คอลัมน์ที่ 2	คอลัมน์ที่ 3
แถวที่ 0	ก้องหล้า [0,0]	แก้วตา [0,1]	แก้มใส [0,2]	กฤษณะ [0,3]
แถวที่ 1	วุฒิชัย [1,0]	วรรณภา [1,1]	วิทยา [1,2]	วราภรณ์ [1,3]
แถวที่ 2	สมปอง [2,0]	สมชาย [2,1]	สมหมาย [2,2]	สงกรานต์ [2,3]

ตัวอย่างเข้าถึงสมาชิกใน Array 2 มิติ

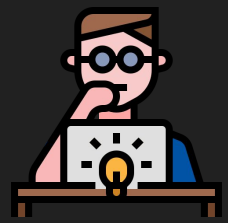
- ชื่อตัวแปร [เลขแถว,เลขคอลัมน์];
- เช่น students [0,1];



การเปลี่ยนแปลงค่าสมาชิกใน Array 2 มิติ

students [0,1] = “สุรสิทธิ์”

students [1,3] = “จักรินทร์”



ขนาด Array ด้วย Length , GetLength

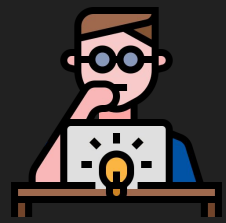
ก้องหล้า	แก้วตา	แก้มใส	กฤษณะ
วุฒิชัย	วรรณภา	วิทยา	วราภรณ์
สมปอง	สมชาย	สมหมาย	สงกรานต์

GetLength (1)

=

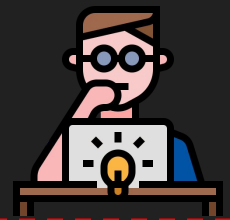
จำนวนคอลัมน์

GetLength (0) = จำนวนแถว



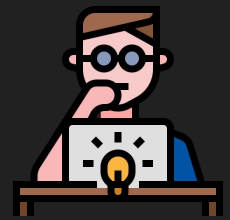
การเข้าถึงข้อมูลด้วย For Loop

```
for (int row = 0; row < students.GetLength(0); row++){
        for (int col=0;col<students.GetLength(1);col++){
            Console.Write(students[row, col]+" ");
        }
        Console.WriteLine();
}
```



การเข้าถึงข้อมูลด้วย ForEach

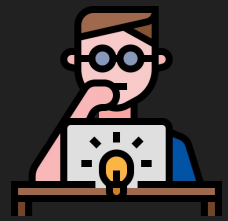
```
foreach(string name in students){
    Console.WriteLine(name);
}
```



Jagged Array

Jagged Array คืออะไร

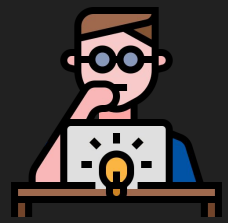
- Array ที่มีข้อมูลสมาชิกภายในเป็น Array (Array ซ้อน Array)
- มีโครงสร้างเป็นรูปแบบแถว (แนวนอน) และคอลัมน์ (แนวตั้ง)
- มีจำนวนสมาชิกในแต่ละ Array ย่อย**เท่ากันหรือต่างกันได้**



การนิยาม Jagged Array

แบบไม่ประกาศค่าเริ่มต้น (ต้องกำหนดจำนวนแถว)

```
string[][] students = new string[4][];
```

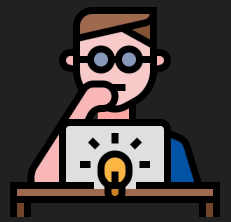


การนิยาม Jagged Array

ประกาศจำนวนสมาชิกในแต่ละแถว

```
string[][] students = new string[4][];

students[0] = new string[3]; // 3 ตัว
students[1] = new string[2];  // 2 ตัว
students[2] = new string[2]; // 2 ตัว
students[3] = new string[1];  // 1 ตัว
```

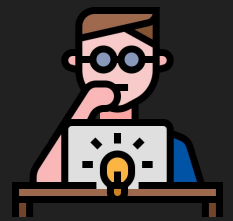


การนิยาม Jagged Array

กำหนดค่าสมาชิกในแต่ละแถว

```
string[][] students = new string[4][];

students[0] = new string[]{“ก้อง”,”แก้ม”,”แก้ว”};
students[1 ] = new string []{“นิก”,”นุ๊ก”};
students[2] = new string[]{“นุ่น”,”นิ่ม”};
students[3] = new string[]{“สมชาย”};
```

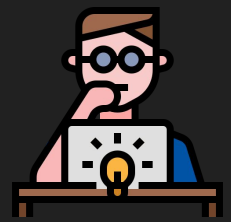


การนิยาม Jagged Array

แบบลดรูป

```
string[][] students = new string[][]{
    new string[]{“ก้อง”,”แก้ม”,”แก้ว”},
    new string []{“นิก”,”นุ๊ก”},
    new string[]{“นุ่น”,”นิ่ม”},
    new string[]{“สมชาย”}

}
```

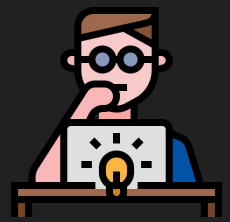


การนิยาม Jagged Array

แบบลดรูป

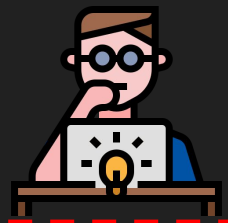
```
string[][] students = {
    new string[]{“ก้อง”,”แก้ม”,”แก้ว”},
    new string []{“นิก”,”นุ๊ก”},
    new string[]{“นุ่น”,”นิ่ม”},
    new string[]{“สมชาย”}

}
```



การเข้าถึงข้อมูลด้วย For Loop

```
for (int row = 0; row < students.Length; row++) {
    for(int col = 0; col< students[row].Length; col++) {
        Console.Write(students[row, col]+" ");
    }
}
```



เมธอด
(Method)

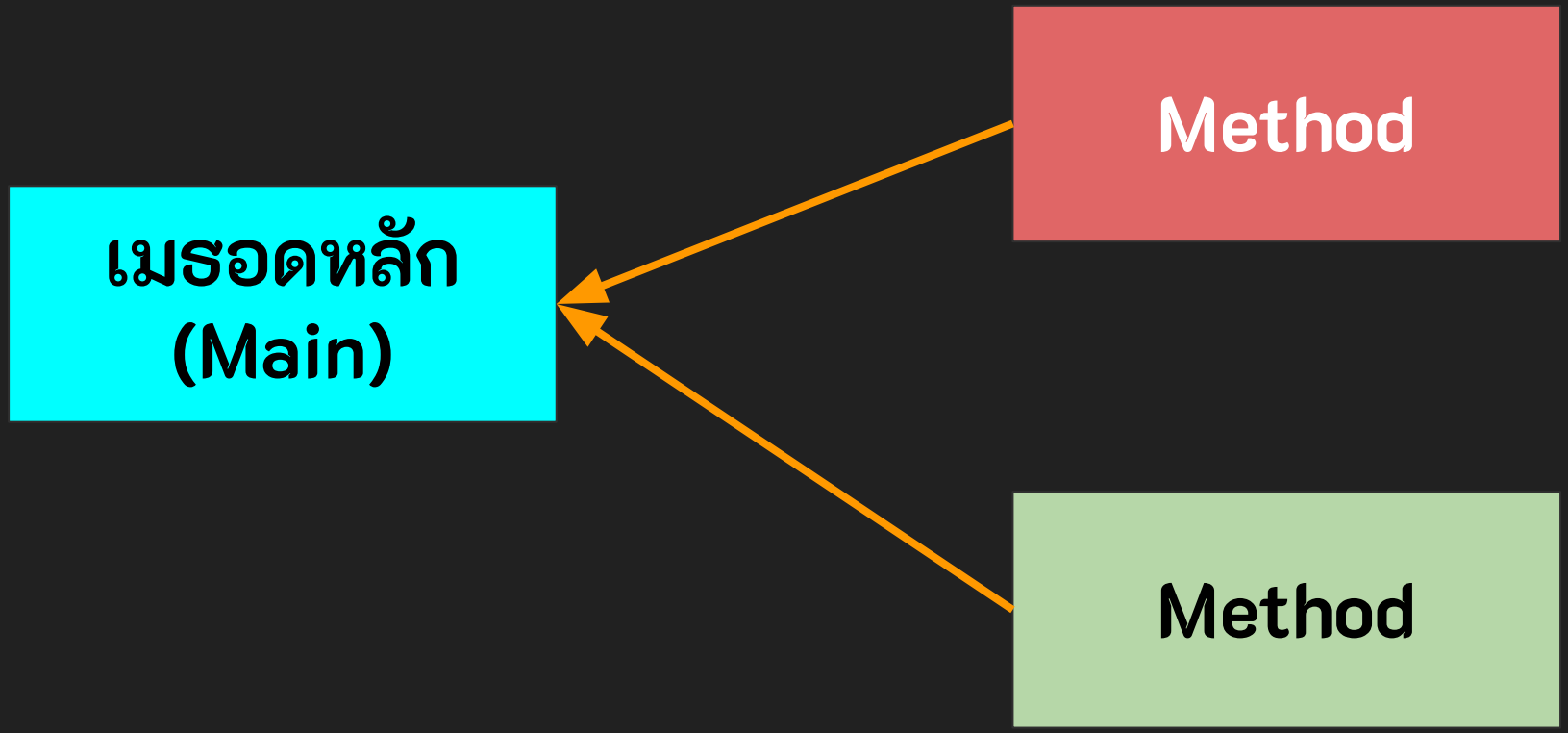
เมธอด (Method) คืออะไร

ชุดคำสั่งที่นำมาเขียนรวมกันเป็นกลุ่มเพื่อให้เรียกใช้งานตามวัตถุประสงค์ที่ต้องการ และลดความซ้ำซ้อนของคำสั่งที่ใช้งานบ่อยๆ เมธอดสามารถนำไปใช้งานได้ทุกที่และแก้ไขได้ในภายหลัง ทำให้โค้ดในโปรแกรมมีระเบียบและใช้งานได้สะดวกมากยิ่งขึ้น

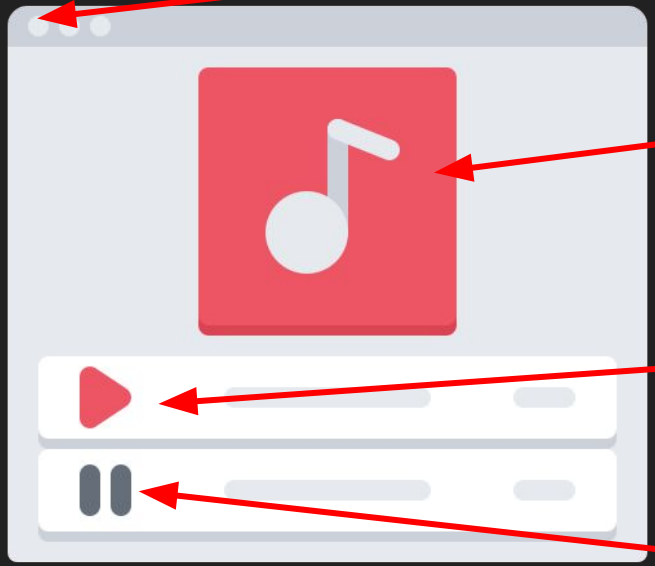
เมธอดมี 2 รูปแบบ คือ

- predefined method - เมธอดที่ทำงานอยู่ใน C#
- user-defined method - เมธอดที่สร้างเอง

เมธอด (Method)



เมธอด (Method)



closeProgram()

ShowMusicName()

playmusic()

pausemusic()

การสร้างเมธอด (Method)

เมื่อสร้างเมธอดในภาษา C# สามารถที่จะเรียกใช้งานได้จากส่วนใดๆ ของโปรแกรมก็ได้ขึ้นกับขอบเขตและระดับการเข้าถึงที่ผู้เขียนได้กำหนดขึ้น

```
type name ( parameter1, parameter2, ... ) {
    statements
}

access_modifier type name ( parameter1, parameter2, ... ) {
    statements
}
```

รูปแบบของเมธอด

1.เมธอดที่ไม่มีการรับและส่งค่า

```
modifier void ชื่อเมธอด(){
    // คำสั่งต่างๆ
}
```

การเรียกใช้งานเมธอด

```
ชื่อเมธอด ();
```

รูปแบบของเมธอด

2.เมธอดที่มีการรับค่าเข้ามาทำงาน

modifier void ชื่อเมธอด(parameter1,parameter2,.....){

// กลุ่มคำสั่งต่างๆ

}

อาร์กิวเมนต์ คือ ตัวแปรหรือค่าที่ต้องการส่งมาให้กับเมธอด (ตัวแปรส่ง)
พารามิเตอร์ คือ ตัวแปรที่เมธอดสร้างไว้สำหรับรับค่าที่จะส่งเข้ามาให้กับเมธอด (ตัวแปรรับ)

การเรียกใช้งานเมธอด

ชื่อเมธอด (argument1,argument2,.....);

รูปแบบของเมธอด (Method)

3.เมธอดที่ส่งค่าออกมาทำงาน

```
modifier type ชื่อเมธอด(){
    return ค่าที่จะส่งออกไป (type)
}
```

รูปแบบของเมธอด

4.เมธอดที่มีการรับค่าเข้ามาและส่งค่าออกไป

```
modifier type ชื่อเมธอด(parameter1,parameter2,.....){
    retrun ค่าที่จะส่งออกไป
}
```

Global Variable & Local Variable

- **Global variable** คือ ตัวแปรที่สามารถเรียกใช้งานได้หลายเมธอดในเวลาเดียวกัน (ตัวแปรของคลาส)
- **Local variable** คือ ตัวแปรสามารถใช้งานได้เฉพาะในบางพื้นที่หรือบางเมธอด (ตัวแปรของเมธอด)

สตรัคเจอร์ (Structure)

ข้อจำกัดของ Array ในกรณีที่มีการเก็บข้อมูลลงไปใน Array สมาชิกทุกตัวที่อยู่ใน Array ต้องมีชนิดข้อมูลเหมือนกัน

แล้วถ้าต้องการอยากเก็บข้อมูลที่มีชนิดข้อมูลต่างกันจะทำอย่างไร ?

สตรัคเจอร์ (Structure)

คือ ข้อมูลแบบโครงสร้างที่นำเอาข้อมูลที่มีชนิดข้อมูลต่างกันมารวบรวมเข้าด้วยกัน แต่มีความสัมพันธ์ของข้อมูลแบบต่อกัน มาเก็บไว้ภายในโครงสร้างเดียวกัน และสมาชิกแต่ละตัวที่อยู่ในสตรัคเจอร์จะเก็บข้อมูลโดยใช้หน่วยความจำแยกกัน

****เปรียบเสมือนกับสร้างชนิดข้อมูลขึ้นมาใช้งานเอง****

การนิยามสตรัคเจอร์

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

ข้อมูลพนักงาน

- ชื่อพนักงาน (string)
- อายุ (int)
- เงินเดือน (double)
- แผนก (string)

เข้าถึงสมาชิกในสตรัคเจอร์

- ชื่อสตรัคเจอร์.สมาชิก

Enum (Enumerator)

Enumerator คือ สิ่งที่ระบุขึ้นเอง หรือหมายถึงตัวแปรที่เป็นรูปแบบตัวเลขจำนวนเต็ม (integer) ที่มีการตั้งชื่อเฉพาะขึ้นมาเพื่อเป็นตัวแทนของกลุ่มข้อมูล

การนิยาม Enum

```
enum ชื่อenum{
    value1,
    value2,
    value3
    ...
}
```

การนิยาม Enum

```
enum Rating{
    VeryBad,
    Bad,
    Good,
    Great,
    Excellent
}
```

จัดการข้อผิดพลาด (Exception)

Exception

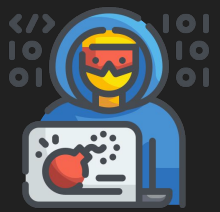
การที่โปรแกรมทำงานบางอย่างแต่เกิดข้อผิดพลาดขึ้นแล้ว
โปรแกรมไม่สามารถจัดการข้อผิดพลาดนั้นได้ ซึ่งทำให้เกิดสิ่งผิดปกติ
หรือ Exception ส่งผลทำให้โปรแกรมหยุดทำงาน

ตัวอย่าง Exception

- ArrayIndexOutOfBoundsException
- ArithmeticException
- ZeroDivisionException
- IOException
- FileNotFoundException
- อื่นๆ

จัดการ Exception ด้วย Try...Catch

```
try{
    // ลองทำคำสั่งในนี้
}catch(Exception){
    // ถ้าเกิดข้อผิดพลาดจะมาทำตรงส่วนนี้
}
```



Try...Catch แบบหลายเหตุการณ์

```
try {
    // ลองทำคำสั่งในนี้
} catch (ExceptionType1) {
    // ถ้าเกิดข้อผิดพลาดที่ 1 จะมาทำตรงส่วนนี้
} catch (ExceptionType2) {
    // ถ้าเกิดข้อผิดพลาดที่ 2 จะมาทำตรงส่วนนี้
}
```

Finally เมื่อเกิดข้อผิดพลาด หรือไม่เกิดก็จะทำงานคำสั่งในส่วนนี้ทุกครั้งคำสั่งที่ระบุมักจะเป็นคำสั่งที่ทำงานส่วนที่สำคัญของโปรแกรม 72

```
try{
   // ลองทำคำสั่งในนี้
}catch(Exception){
    // ถ้าเกิดข้อผิดพลาดจะมาทำตรงส่วนนี้
}finally {
      // คำสั่งต่างๆ
}
```