

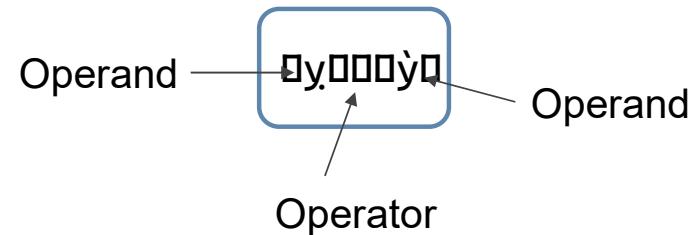


Operators

Operator

□ Operator

- Some special symbol that tells **compiler** to perform some action on operands



□ C# supports:

- Unary operators:
 - Requires **one** operand such as `x++`
- Binary operators:
 - Requires **two** operands in the expression such as `x + 2`
- Ternary operators:
 - Requires **three** operands such as Conditional (? :) operator.

Arithmetic Operators

Operator	Description	Example (y=5)	Result
+	Addition	x=y+2	x=7
-	Subtraction	x=y-2	x=3
*	Multiplication	x=y*2	x=10
/	Division	x=y/2	x=2.5
%	Modulus (division remainder)	x=y%2	x=1
++	Increment (postfix, prefix)	x=++y	x=6
--	Decrement (postfix, prefix)	x= - -y	x=4

```
int x,y;  
x=y=10;  
Console.WriteLine(++x); // print 11  
Console.WriteLine(y++); // print 10
```

Assignment Operators

Operator	Example	Same As	Result (y=10)
=	x=y		x=5
+=	x+=y	x=x+y	x=15
-=	x-=y	x=x-y	x=5
=	x=y	x=x*y	x=50
/=	x/=y	x=x/y	x=2
%=	x%=y	x=x%y	x=0

Bitwise Operators

Operator	Description
&	Bitwise AND
	Bitwise OR
^	Bitwise XOR
~	Bitwise NOT
<<	Bitwise Left Shift
>>	Bitwise Right Shift

```
int a = 60; /* 60 = 0011 1100 */
int b = 13; /* 13 = 0000 1101 */
int c = 0;

c = a & b; /* 12 = 0000 1100 */
Console.WriteLine("Line 1 - Value of c is {0}", c);

c = a | b; /* 61 = 0011 1101 */
Console.WriteLine("Line 2 - Value of c is {0}", c);

c = a ^ b; /* 49 = 0011 0001 */
Console.WriteLine("Line 3 - Value of c is {0}", c);

c = ~a;    /* -61 = 1100 0011 */
Console.WriteLine("Line 4 - Value of c is {0}", c);

c = a << 2; /* 240 = 1111 0000 */
Console.WriteLine("Line 5 - Value of c is {0}", c);

c = a >> 2; /* 15 = 0000 1111 */
Console.WriteLine("Line 6 - Value of c is {0}", c);
```

Comparison Operators

Operator	Description
<	Less than
>	Greater than
<=	Less than or equal to
>=	Greater than or equal to
==	Equality
!=	Inequality

Logical Operators

Operator	Description
&&	Logical "AND" – returns true when both operands are true; otherwise it returns false
	Logical "OR" – returns true if either operand is true. It only returns false when both operands are false
!	Logical "NOT"—returns true if the operand is false and false if the operand is true. This is a unary operator and precedes the operand

Precedence and Associativity

- **Operator Precedence:** Determines the order in which operators are evaluated. Operators with higher precedence are evaluated first.
- **Operator Associativity:** Determines the order in which operators of the same precedence are processed.
 - In an expression with multiple operators, the operators with higher precedence are evaluated before the operators with lower precedence
 - Ex:

ઇન્જેશન્સ એન્ડ ટેકનોલોજીસ, મારું, મારું, મારું
Công ty Cổ phần Công nghệ Maru, Maru, Maru, Maru

ઇન્જેશન્સ એન્ડ ટેકનોલોજીસ, મારું, મારું, મારું
Công ty Cổ phần Công nghệ Maru, Maru, Maru, Maru

Precedence and Associativity

□ Precedence from high to low

Category	Operator	Associativity
Postfix	(), [], ++, --	Left To Right
unary	--, ++, !	Right to Left
mutilation	* , / , %	Left to Right
Addition	+ , -	Left to Right
Shift	>>, <<	Left to Right
Relational	<, <=, >, >=	Left to Right
Equality	==, !=	Left to Right
Logical and	&&	Left to Right
Logical Or		Left to Right
Conditional	?:	Right to Left
Assignment	=, +=, -=, *=, /=, %=	Right to Left

Assignment

- Get sum , average for 2 numbers entered by the user



Control Statements

Controlling Program Flow

- Control Statements that can be used are:
- **Conditional Statements**
 - ifelse
 - switch/case
 - Trinary operator ?:
- **Loop Statements**
 - for
 - while
 - do...while

if and if ... else

- Used for check certain value or range of values

```
if (condition)
{
    do something;
}
```

```
if (condition)
{
    do something;
}
else
{
    do something else;
}
```

îŋɟ̥ə̄gsá̄dē̄
Cô̄ŋşȫl̥ê̄ñ̥s̥î̄t̥j̥ê̄l̥î̄n̥ê̄ñ̥É̥ŋ̥t̥ê̄ñ̥s̥î̄g̥sá̄dē̄ñ̥
st̥s̥î̄n̥j̥ñ̥s̥ñ̥ñ̥Cô̄ŋşȫl̥ê̄ñ̥R̥ê̄ñ̥d̥l̥î̄n̥ê̄ñ̥
g̥sá̄dē̄ñ̥ñ̥î̄ŋ̥t̥ñ̥R̥ă̄s̥s̥ê̄ñ̥ñ̥
īg̥ñ̥ñ̥g̥sá̄dē̄ñ̥ñ̥` . ñ̥
ñ̥
ñ̥Cô̄ŋşȫl̥ê̄ñ̥s̥î̄t̥j̥ê̄l̥î̄n̥ê̄ñ̥R̥ă̄s̥s̥ñ̥ñ̥
ñ̥
ê̄l̥ş̥ê̄
ñ̥
ñ̥Cô̄ŋşȫl̥ê̄ñ̥s̥î̄t̥j̥ê̄l̥î̄n̥ê̄ñ̥G̥ă̄ñ̥l̥ñ̥ñ̥
ñ̥

switch ... case

- Used for multiple certain values
 - Ex: check for value 1 ,2, other values

```
switch (expression)
{
    case value1:
        statements
        break;
    case value2:
        statements
        break;
    default :
        statements
}
```

şxítşçhôôğsâđêô
ôô
ôôôçâşêô , ôôôôôôôôôôôôôôôôôôôôôôôô
ôôôôôCôñşôl'êôôôsîtşêL'îñêôôÔñjêôôô
ôôôçséâlô
ôôôçâşêô , ô
ôôôôôCôñşôl'êôôôsîtşêL'îñêôôôôxôôô
ôôôçséâlô
ôôôđéğâül'ýô
ôôôôôCôñşôl'êôôôsîtşêL'îñêôôôôtşhêôôââl'ûêôô
ôôôçséâlô
ôô

Ternary operator ? :

çôηđîťîôη𠀠𠀠𠀠çôηşêruêηt𠀠𠀠𠀠âl'ťêseηâťîwê

îηt̄y𠀠𠀠, .𠀠

st̄sîn̄g𠀠w𠀠

îg𠀠y𠀠, .𠀠

𠀠

𠀠𠀠𠀠â𠀠𠀠𠀠, .𠀠𠀠

𠀠

êl'sê

𠀠

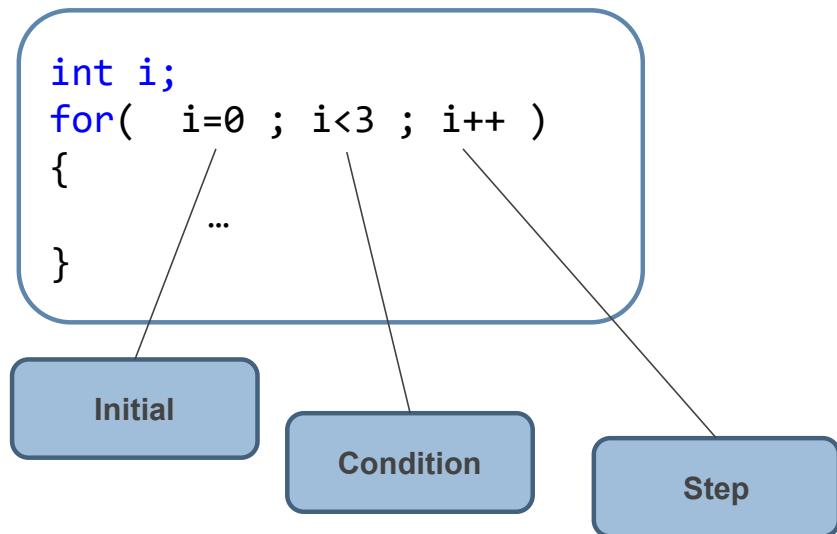
𠀠𠀠𠀠â𠀠𠀠𠀠âl'ťhêseNnučêseN

𠀠

â𠀠𠀠𠀠y𠀠𠀠𠀠, .𠀠𠀠𠀠, .𠀠𠀠𠀠âl'ťhêseNnučêseN

for

- Used for known number of loops



ກົດເນື້ອງໃຫຍ່ເປົ້າມາ . ໂັດໃຫຍ່, ໂັດໃຫຍ່
ໂນ
ໂນໂຄງສໍ່ລໍາວິວສີ່ຈະລຶງເນັ້ນ . ໂພນ້າໃຫຍ່
ໂນ

ກົດເນື້ອງໃຫຍ່ເປົ້າມາ , ໂັດໃຫຍ່ . ໂັດໃຫຍ່
ໂນ
ໂນໂຄງສໍ່ລໍາວິວສີ່ຈະລຶງເນັ້ນ . ໂພນ້າໃຫຍ່
ໂນ

while

- Used when number of loops is unknown but depends on certain condition

```
while(x<5)
{
    ...
}
```

ન્યુયોર્ક, અ
ખિલેન્ડાયર્ડ, અ
અ
અમેરિકાની વિશેષ સિદ્ધેની નીચે અન્યથા
અન્યથા
અ

do... while

- Same as while but loops at least one time

```
do  
{  
    ...  
} while (x < 6);
```

loops flow control

- **break** statement :

- The break statement will terminate looping and continue executing the code that follows after the loop (if any).

- **continue** statement:

- The continue statement will terminate the current loop and continue with the next loop.

გოს ინტერი, მას დაუკავშირდეთ
□
მაგრავი, . .
მაჩეალ
მაცესილე წილელი ენერგია. მაგრავი
□

გოს ინტერი, მას დაუკავშირდეთ
□
მაგრავი, . .
მარიაცენტრუებ
მაცესილე წილელი ენერგია. მაგრავი
□

Assignment

- Take two integer from user and get max of them
- Create simple menu and get user selection from it
 - To calculate sum or get max or get min
- Simple calculator
- ***Optional Assignment:***
 - Magic Box

Assignment

Magic Box

- Row -- Column --
- Row++

6	1	8
7	5	3
2	9	4



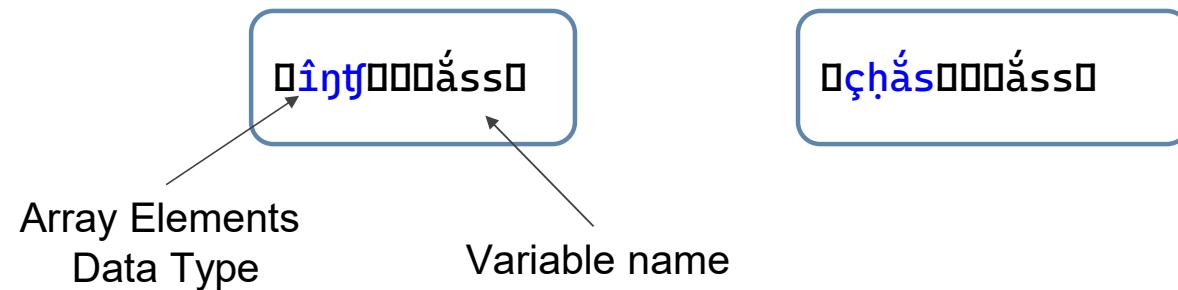
Arrays

Arrays

- ◻ Array is reference type data type
- ◻ Used to store collection of homogenous elements (same data type)
- ◻ Number of elements in array could be dynamically allocated but once the array is allocated its size could not be changed directly

Single Dimension Array

- Declare a reference to single dimension array

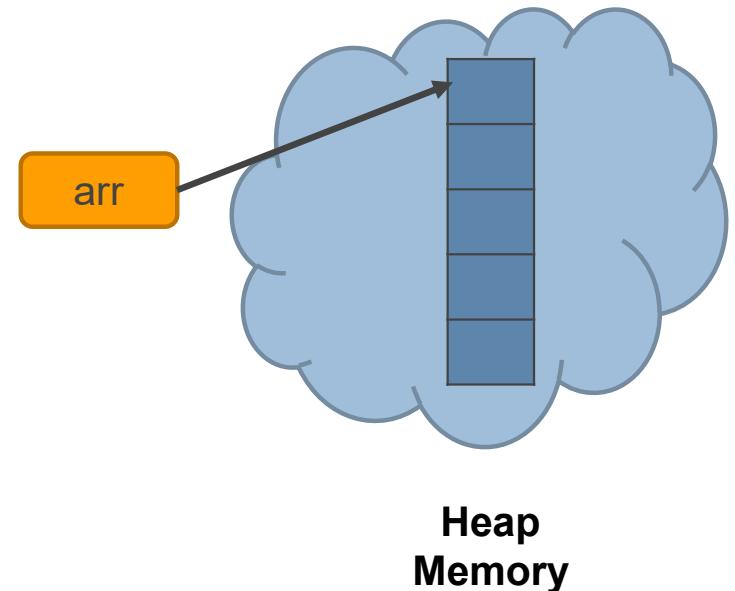
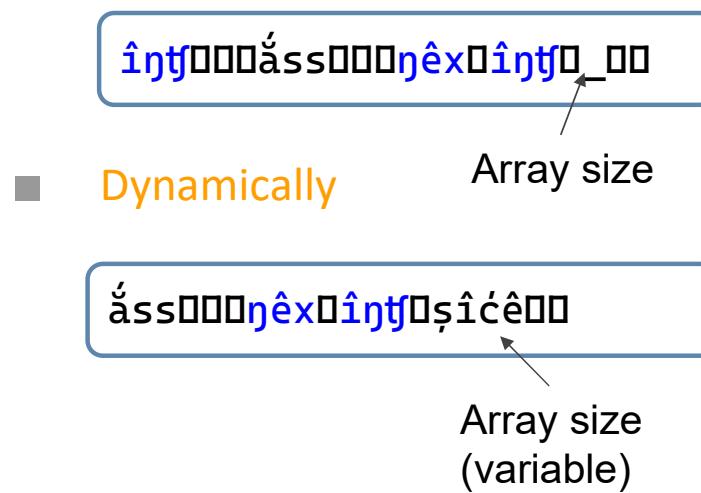


Single Dimension Array

Initialization of array reference

Explicitly

- Array elements auto initialized with default values(0, false, null)
- Statically

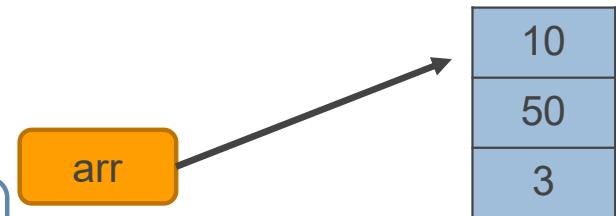


Single Dimension Array

- Initialization of array reference

- Implicitly* (Array initializer)

```
int[] arr = new int[] { 10, 50, 3 };  
int[] arr = { 10, 50, 3 };
```

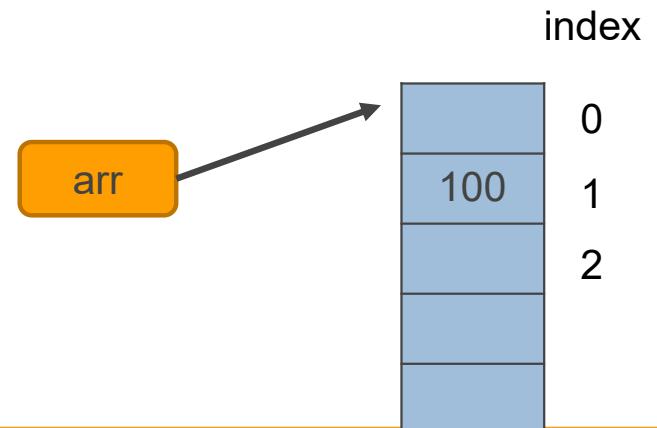


- Accessing array elements

- Array elements could be accessed through index (starts with 0)

```
arr[1]=100;
```

index



Single Dimension Array

- Using for loop with single dimension array
 - *for* Loop

ກົດສອບຕີ່ຈະມາດີວິນດີ່



ມາດີວິນດີ່ຈະມາດີວິນດີ່



- *foreach* loop
 - Used for read only

ກົດສອບຕີ່ຈະມາດີວິນດີ່



ມາດີວິນດີ່ຈະມາດີວິນດີ່



Assignment

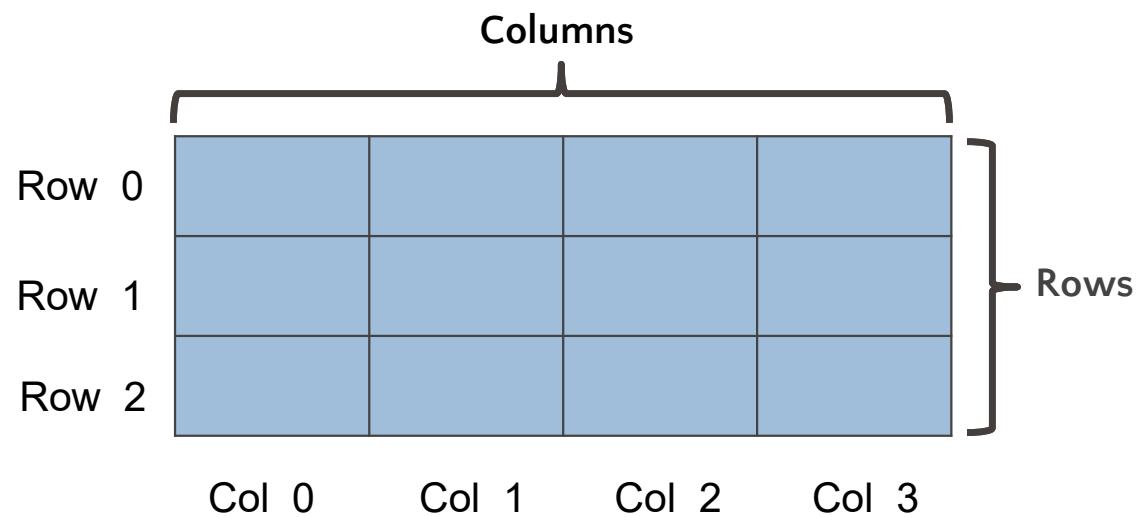
- Get sum, average ,max ,min of integers given by the user
 - Let the user determine number of integers
- Calculate the result of one operation Equation
 - Ex: user Input $5*3 \rightarrow$ result 15
 - Method used (string)
 - Contains
 - Split



```
if __name__ == "__main__":
    print("Hello, world!")
```

Multi Dimensional Array

- 2-dimension array



Multi Dimensional Array

- Declare reference to multi-dimensional array

`int[][] arr`

`int[] arr`

- Initialization reference

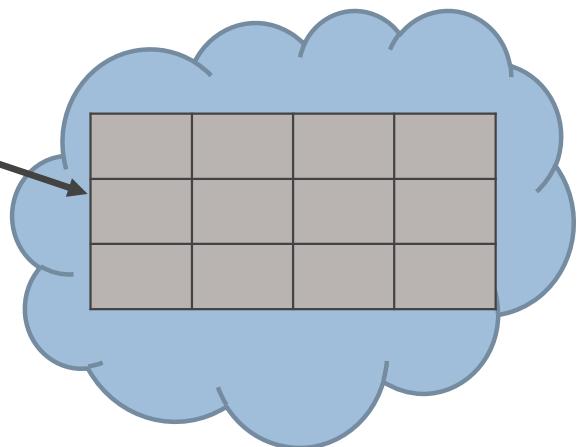
- *Explicitly*

```
arr = new int[3, 4];
// 3 rows, 4 columns
```

- *implicitly*

```
int[,] arr = new int[,] {
    {1,2,3},
    {3,4,5}
}; // 2 rows, 3 columns
```

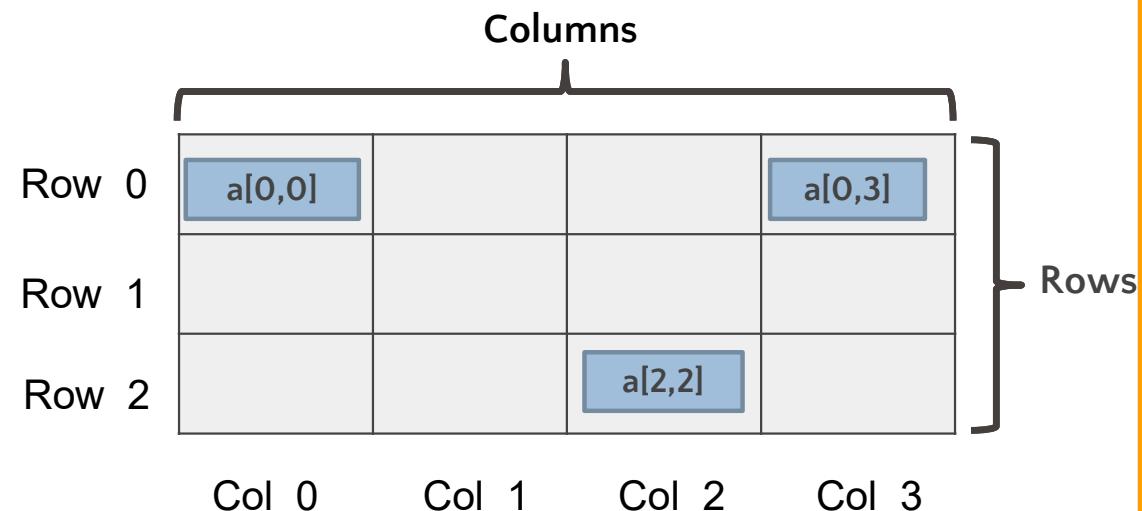
arr



Multi Dimensional Array

- Access array elements
 - Through 2 indices

```
arr[0,3]=10;
```



Multi Dimensional Array

- Looping through 2-D Arrays
 - Using 2 nested for loop

```
for(int j=0 ; j<3 ; j++)  
{  
    for(int i=0 ; i<4 ; i++)  
    {  
        Console.WriteLine (arr[ j ,i]);  
    }  
}
```

0,0	0,1	0,2	0,3
1,0	1,1	1,2	1,3
2,0	2,1	2,2	2,3

Multi Dimensional Array

- Array properties

- Length → number of the array element

- Array Methods

- Static Methods

- Sort
 - BinarySearch
 - Reverse

```
arr = new int[]{5,7,2};  
Array.Sort(arr); // Static Method
```

- instance method

- GetLength(int dimension)
 - Gets number of elements in certain dimension

```
int[,] arr = new int[,] { {1,2,3}, {3,4,5} };  
arr.GetLength(0); // 2  
arr.GetLength(1); // 3
```

Assignment

- Design a program to Get the degree of 3 student with 4 subject from user input then calculate
 - The sum of marks for each student
 - The average for each subject