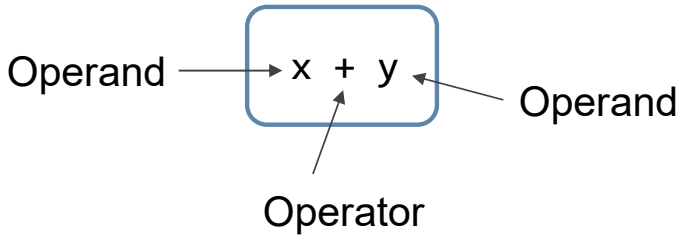
Operators

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:

```
int a = 2 + 2 * 2;
Console.WriteLine(a); // 6
```

```
int a = (2 + 2) * 2;
Console.WriteLine(a); // 8
```

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
  - if ….else
  - 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;
}
```

```
int grade;
Console.WriteLine("Enter grade");
string s = Console.ReadLine();
grade = int.Parse(s);
if (grade >= 60)
 {
  Console.WriteLine("Pass");
 }
else
 {
  Console.WriteLine("Fail");
 }
```

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
}
```

```
switch (grade)
 {
   case 1:
     Console.WriteLine("One");
   break;
   case 2:
     Console.WriteLine("two");
   break;
   default:
     Console.WriteLine("other value");
   break;
 }
```

Ternary operator ? :

```
condition ? consequent : alternative
```

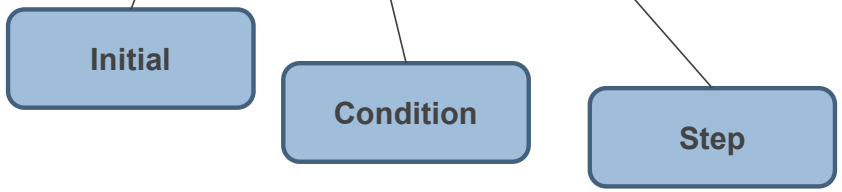
```
int x = 10;
string v;
if (x==10)
{
    v = "10";
}
else
{
    v = "Other Number";
}
```

```
v = (x == 10) ? "10" : "Other Number";
```

for

- Used for known number of loops

```
int i;
for(  i=0 ; i<3 ; i++ )
{
        …
}
```




```
for (int i = 0; i < 3; i++)
 {
   Console.WriteLine("{0}", i);
 }
```

```
for (int i = 2; i >=0; i--)
 {
   Console.WriteLine("{0}", i);
 }
```

while

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

```
while(x<5)
{
        …
}
```

```
int y = 10;
while (y < 20)
 {
   Console.WriteLine("{0}", y);
   y++;
 }
```

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.

```
for (int i = 2; i >= 0; i--)
{
  …
  if (x == 10)
   break;
  Console.WriteLine("{0}", i);
 }
```

```
for (int i = 2; i >= 0; i--)
 {
       …
   if (x == 10)
     continue;
   Console.WriteLine("{0}", i);
 }
```

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

```
int[] arr;
```

```
char[] arr;
```

Single Dimension Array

- Initialization of array reference
  - ***Explicitly***
    - Array elements auto initialized with default values(0, false, null)
    - Statically

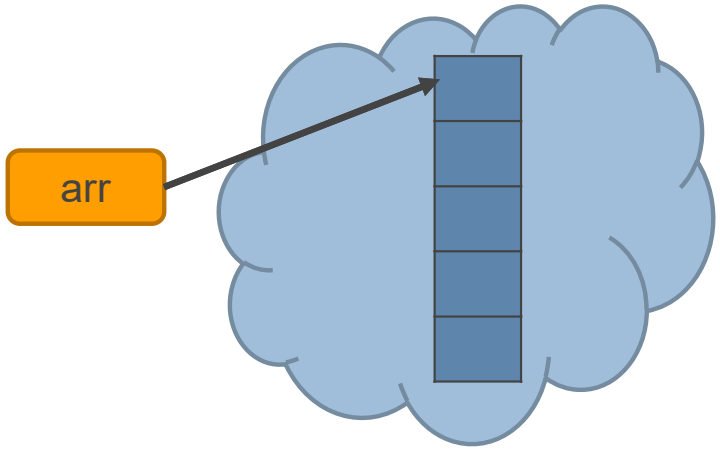
```
int[] arr = new int[5];
```

Array size

    - Dynamically

```
arr = new int[size];
```

Array size
(variable)

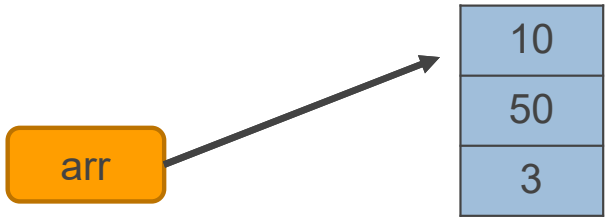


Heap Memory

Single Dimension Array

- Initialization of array reference
  - ***Implicitly*** (Array initializer)

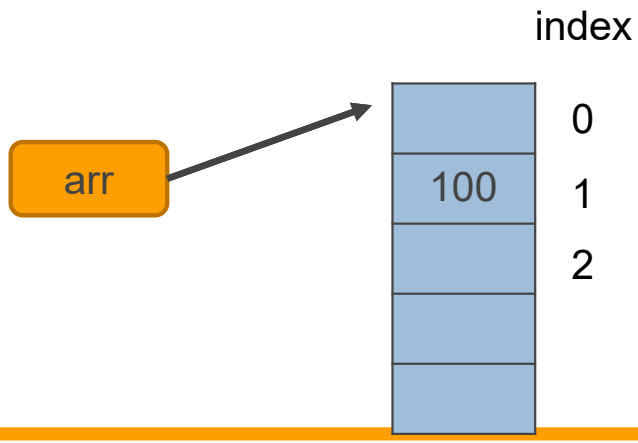
```
int[] arr = new int[] { 10, 50, 3 };
int[] arr = { 10, 50, 3 };
```



- 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

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

  - *foreach* loop
    - Used for read only

```
foreach (int x in arr)
{
 Console.WriteLine("{0}", x );
}
```

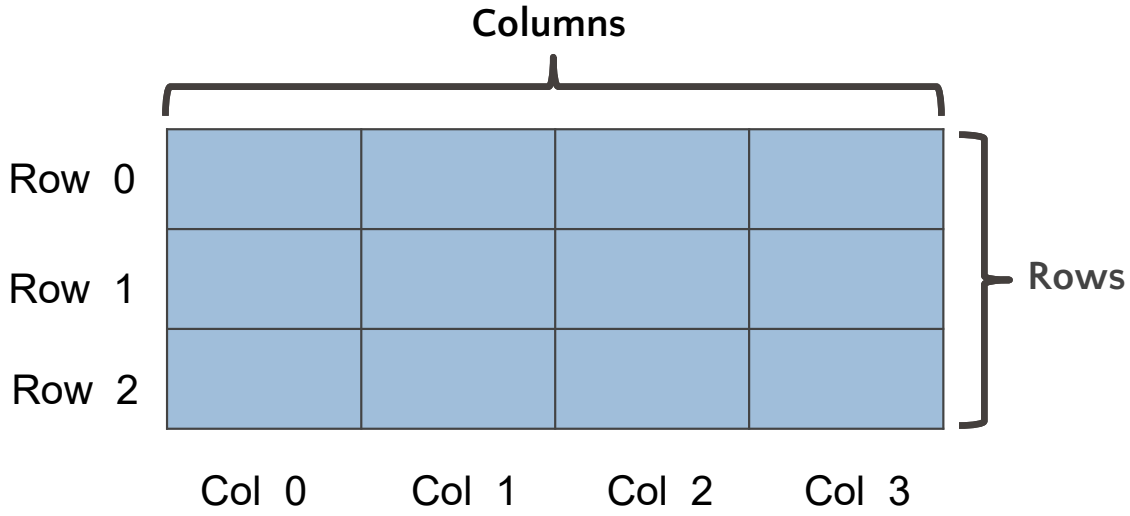
Assignment

- 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 ➔ result 15
  - Method used (string)
    - Contains
    - Split

```
5+3
8
6*5
30
```

Multi Dimensional Array

- 2-dimension array



Multi Dimensional Array

- Declare reference to multi-dimensional array

```
int[,] arr;
```

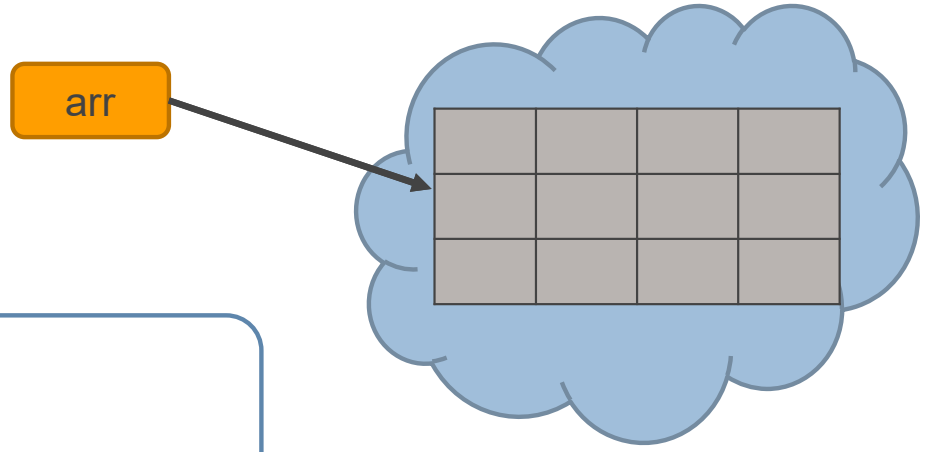
```
int[,,] arr;
```

- Initialization reference
  - *Explicitly*

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

  - *implicitly*

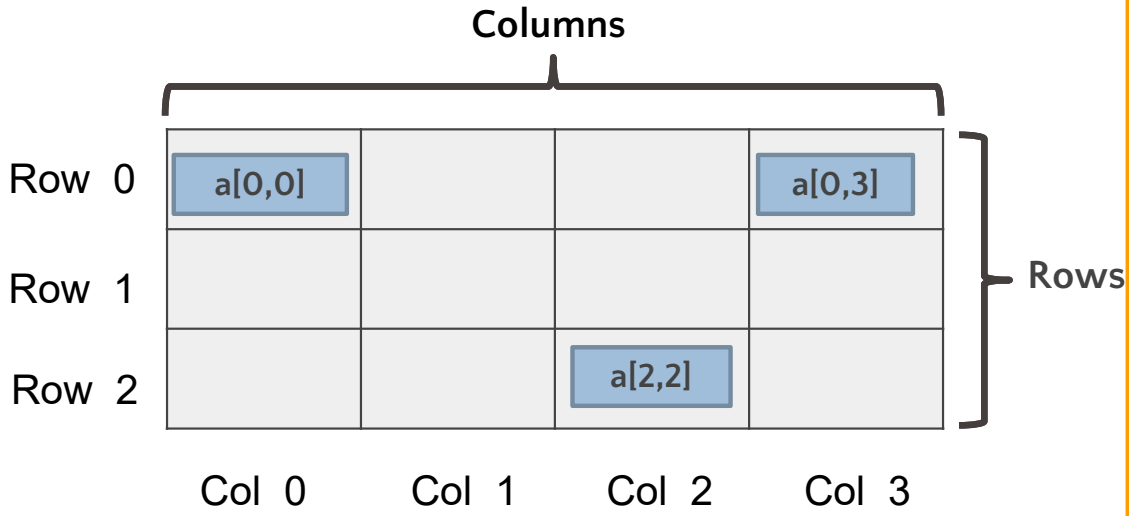
```
int[,] arr = new int[,] {
                {1,2,3},
                {3,4,5}
                }; // 2 rows, 3 columns
```



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

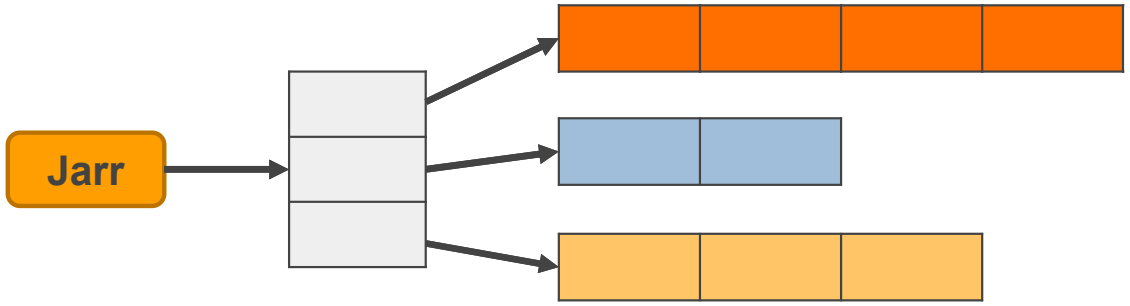
```
int[,] arr = new int[,] { {1,2,3}, {3,4,5} };
arr. GetLength (0); // 2
arr. GetLength (1); // 3
```

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

Multi Dimensional Array

- Jagged Array (Array of Arrays)



- Declare Jagged Array Reference

```
int[][]Jarr ;
```

Multi Dimensional Array

- Initialization reference

```
int[][] jarr = new int[3][];
jarr[0] = new int[4] { 1, 2, 3 ,4 };
jarr[1] = new int[2] { 4, 5};
jarr[2] = new int[3] { 10, 15, 20};
```

  - Using array initializer

```
int[][] jArray = new int[][] {
          new int[] { 1, 2, 3 ,4 },
          new int[] { 4, 5},
          new int[] { 10, 15, 20}
            };
```

Assignment

- Design a program that get from user input
  - Number of class room
  - Number of student in each class
  - Mark for each student
- Then calculate the
  - Average mark for each class room