

**1\_1 : Store & Display values in different variable of different type  
(Integer, Double, Float, Long, Short, Byte, Char, Boolean, String)**

**CODE:**

```
fun main() {  
    val intval: Int = 20  
    val floatval: Double = 2.0  
    val charval: Char = 'p'  
    val stringval: String = "praveen"  
    val boolval: Boolean = true  
    val doubleval: Double = 95.7  
    val longval: Long = 7799643281  
    val shortval: Short = 1  
    val byteval: Byte = 64  
  
    println("Integer Value : " + intval)  
    println("Float Value : " + floatval)  
    println("Character Value : " + charval)  
    println("String Value : " + stringval)  
    println("Boolean Value : " + boolval)  
    println("Double Value : " + doubleval)  
    println("Long Value : " + longval)  
    println("Short Value : " + shortval)  
    println("Byte Value : " + byteval)  
  
}
```

**OUTPUT:**

```
"C:\Users\kallam venugopal\.jdk\openjdk-18.0.2.1\bin\java.exe" "-javaa
Integer Value : 20
Float Value : 2.0
Character Value : p
String Value : praveen
Boolean Value : true
Double Value : 95.7
Long Value : 938145520
Short Value : 1
Byte Value : 64

Process finished with exit code 0
```

## 1\_2.Type conversion:

**Integer to Double, String to Integer, String to Double.**

### CODE:

```
fun main(){
    var i:Int = 14
    println("Integer value: $i")
    var d:Double = i.toDouble()
    println("Double Value (From Integer):$d")
    var s:String = "10"
    println("String Value: $s")
    var i1:Int = s.toInt()
    println("Integer Value1 (From String):$i1")
    var d1:Double = s.toDouble()
    println("Double Value (From String):$d1")
}
```

### OUTPUT:

```
"C:\Users\kallam venugopal\.jdk\openjdk-18.0.2.1\bin\java.exe" "-ja
Integer value: 14
Double Value (From Integer):14.0
String Value: 10
Integer Value1 (From String):10
Double Value (From String):10.0

Process finished with exit code 0
```

**1\_3.Scan student's information and display all the data****CODE:**

```
fun main() {  
    print("Student Enrollment No.: ")  
    var sn: Long = readLine()!!.toLong()  
    print("Student Name: ")  
    var sname = readLine()  
    print("Student Branch: ")  
    var sb = readLine()  
    print("Student College Name: ")  
    var scn = readLine()  
    print("Student University Name: ")  
    var sun = readLine()  
    print("Student Age: ")  
    var sa: Int = readLine()!!.toInt()  
    println("*****")  
    println("Student Enrollment No.: $sn")  
    println("Student Name: $sname")  
    println("Student Branch: $sb")  
    println("Student College Name: $scn")  
    println("Student University Name: $sun")  
    println("Student Age: $sa")  
}
```

**OUTPUT:**

```
Student Enrollment No.: 20012531037
Student Name: praveen
Student Branch: ce_ai
Student College Name: uvpce
Student University Name: ganpat university
Student Age: 20
*****
Student Enrollment No.: 20012531037
Student Name: praveen
Student Branch: ce_ai
Student College Name: uvpce
Student University Name: ganpat university
Student Age: 20

Process finished with exit code 0
```

#### 1\_4.Find the number is odd or even by using Control Flow inside println() method

##### CODE:

```
fun main(){
    print("Enter number : ")
    var x:Int = readLine()!!.toInt()
    if(x % 2 == 0)
        println("Even")
    else
        println("Odd")
}
```

Output:

```
Enter number : 20
Even

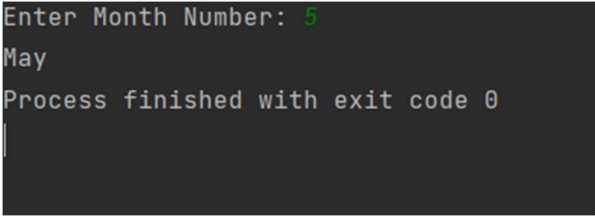
Process finished with exit code 0
```

#### 1\_5.Display month name using When

##### CODE:

```
fun main(){
```

```
print("Enter Month Number: ")
var m:Int = readln()!!.toInt()
when(m){
    1 -> print("January")
    2 -> print("February")
    3 -> print("March")
    4 -> print("April")
    5 -> print("May")
    6 -> print("June")
    7 -> print("July")
    8 -> print("August")
    9 -> print("September")
    10 -> print("October")
    11 -> print("Nuvember")
    12 -> print("December")
    else -> print("Enter proper number")
}
}
```

**OUTPUT:**

```
Enter Month Number: 5
May
Process finished with exit code 0
```

**1\_6. By using a user defined function perform all arithmetic operations.**

**CODE:**

```
fun main(){

    fun add(vararg x:Int){

        var add:Int = 0

        for(i in x){
```

```
        add += i
    }

    var y:String = ""

    for(i in x){
        y += "$i,"
    }

    println("Addition of $y is $add")
}

fun sub(vararg x:Int){
    var sub:Int = 0
    var flag = 0
    for(i in x){
        if (flag == 0){
            flag = 1
            continue
        }

        sub -= i
    }

    var y:String = ""

    for(i in x){
        y += "$i,"
    }

    println("Subtraction of $y is $sub")
}

fun mul(vararg x:Int){
    var mul:Int = 1
    for(i in x){
        mul *= i
    }
}
```

```
}

var y:String = ""

for(i in x){

    y += "$i,"

}

println("Multiplication of $y is $mul")

}

fun div(vararg x:Int) {

    var div:Double = x[0].toDouble()

    for (i in x) {

        if(i == x[0]) {

            continue

        }

        div /= i

    }

    var y: String = ""

    for (i in x) {

        y += "$i,"

    }

    println("Division of $y is $div")

}

add(1,1,1,1)

sub(0,1,1,5)

mul(2,2,2,2)

div(20,2,2,2)

}
```

**OUTPUT:**

```
Addition of 1,1,1,1, is 4
Subtraction of 0,1,1,5, is -7
Multiplication of 2,2,2,2, is 16
Division of 20,2,2,2, is 2.5

Process finished with exit code 0
```

**1\_7.Find the factorial of number by recursion. Explain "tailrec" keyword.**

**CODE:**

```
fun main() {
    print("Enter a number you want it's factorial: ")
    val n = readLine()!!.toInt()
    println("Factorial using Recursion " + recursionFact(n))
    println("Factorial using tailrec " + fact(n))
}
fun recursionFact(n: Int): Int {
    if (n == 1 || n == 0) {
        return 1
    }
    return n * recursionFact(n - 1)
}
tailrec fun fact(n: Int, temp: Int = 1): Int {
    return if (n == 1) {
        temp
    } else {
        fact(n - 1, temp * n)
    }
}
```

**OUTPUT:**

```
Enter a number you want it's factorial: 10
Factorial using Recursion 3628800
Factorial using tailrec 3628800

Process finished with exit code 0
```



**1\_8.Create different types of Array as shown in image. Explore Arrays.deepToString(), contentDeepToString() methods, IntArray variable .joinToString() and use in program to print Array. Explore range, downTo, until etc. for loop and use in this program. Sort Array of Integer data type without using inbuilt function & with using inbuilt function**

**CODE:**

```
fun main() {    val
myArray1 = arrayOf(12, 13,
15, 16)    println("Using
arrayOf method: ")
    println(myArray1.contentToString())

    val myArray2 =
Array<Int>(5){0}
println("Using Array<>() method:
")
println(myArray2.contentToString()
)

    val myArray3 = Array(5, { i -> i * 2
})    println("Using Array<>() and lambda
function: ")
println(myArray3.contentToString())

    val myArray4 =
intArrayOf(5,10,20,12,15)
println("Using intArrayOf() : ")
    println(myArray4.contentToString())

    val myArray5 = arrayOf(intArrayOf(2, 3),
intArrayOf(5, 6))    println("2D array :")
println(myArray5.contentDeepToString())

    println("Array
Sorting...")
    var arr =
Array<Int>(5){0}
    var x:Int = 0
    println("Enter the
values: ")
    while (x <
arr.size)
    {
        print("arr[$x] : ")
        arr[x] =
Integer.valueOf(readLine())
        x++    }
    println(arr.contentToString())
```

```

        println("***** With In-Built Function
        *****")    println("Before Sorting : ")
        println(arr.contentToString())

        arr.sort()
        println("After
        Sorting : ")
        println(arr.content
        ToString())
        not()

    }
    fun not(){        var
    arr1 =
    Array<Int>(5){0}
    var x:Int = 0
    var a:Int        var
    b:Int
    println("Enter the
    values: ")
    while (x <
    arr1.size)
        {        print("arr[$x] :
    ")        arr1[x] =
    Integer.valueOf(readLine())
    x++        }
        println(arr1.contentToString())

        println("***** Without In-Built
        Function *****")    println("Before
        Sorting : ")
        println(arr1.contentToString())
        println("After Sorting : ")        var
        temp: Int        for(a in arr1.indices){
        for(b in (a+1) until arr1.size){
        if (arr1[a] > arr1[b]){
        temp = arr1[a]
        arr1[a] = arr1[b]
        arr1[b] = temp
        }

        }
        }
        println(arr1.contentToString())
    }

```

**OUTPUT:**

```

Using array() method:
[8, 12, 14, 18]
Using Array<>() method:
[0, 0, 0, 0, 0]
Using Array<>() and lambda function:
[0, 2, 4, 6, 8]
Using intArrayOf() :
[7, 10, 18, 12, 15]
2D array :
[[2, 3], [5, 6]]
Array Sorting...
Enter the values:
arr[0] : 4
arr[1] : 2
arr[2] : 1
arr[3] : 5
arr[4] : 3
[4, 2, 1, 5, 3]
***** With In-Built Function *****
Before Sorting :
[4, 2, 1, 5, 3]
After Sorting :
[1, 2, 3, 4, 5]

```

## 1\_9.Find the max number from ArrayList.

### CODE:

```

fun main(){    var
arr =
Array<Int>(5){0}
var x:Int = 0
println("Enter the
values: ")
while (x <
arr.size)
{
    print("arr[$x] : ")
arr[x] =
Integer.valueOf(readLine())
x++    }

println(arr.contentToString())
arr.sort()    val
y: Int = arr[4]
println("The largest
Number is : $y")
}

```

**OUTPUT:**

```
Enter the values:
arr[0] : 25
arr[1] : 20
arr[2] : 30
arr[3] : 10
arr[4] : 5
[25, 20, 30, 10, 5]
The largest Number is : 30

Process finished with exit code 0
```

**1\_10. Write Different types of Class & Constructor. Create a class Car and set various members like type, model, price, owner, milesDrive. add the function getCarPrice in it. Create an object of Car class and access property of it. (getCarInformation(), getOriginalCarPrice(), getCurrentCarPrice(), displayCarInfo() etc.)**

**CODE:**

```
fun main() {
    val car1 = Car("Lambo, 2018", "Anish", 115, 100000.0, 98950.0)
    car1.getCarFullDetails()
    val car2 = Car("BMW, 2019", "malpani", 120, 400000.0, 399800.0)
    car2.getCarFullDetails()
    val Cars = ArrayList<Car> (2)
    val car3 = Car("Audi, 2017", "vikas", 50, 1080000.0, 1079000.0)
    val car4 = Car("Maruti, 2020", "parth", 20, 4000000.0, 3998000.0)
    Cars.add(car3)
    Cars.add(car4)
    for (i in Cars){
        println("-----")
        i.getCarFullDetails()
    }
}

class Car(private val model: String, private val owner: String, private val miles: Int, private val original: Double, private val current: Double) {
```

```
init {  
    println("Object of class is Created and Init is Called.")  
}  
private fun info(): String {  
    return model  
}  
private fun carowner(): String {  
    return owner  
}  
private fun milesDrive(): Int {  
    return miles  
}  
private fun orgprice(): Double {return original  
}  
private fun currprice(): Double {  
    return current  
}  
fun getCarFullDetails() {  
    println("-----")  
    println("Car Information : ${info()}")  
    println("Car owner : ${carowner()}")  
    println("Miles Drive : ${milesDrive()}")  
    println("Original Car Price : ${orgprice()}")  
    println("Current Car Price : ${currprice()}")  
    println("-----\n")  
}  
}
```

**OUTPUT:**

```
Car Information : Lambo, 2018
Car owner : Anish
Miles Drive : 115
Original Car Price : 100000.0
Current Car Price : 98950.0
-----

Object of class is Created and Init is Called.
-----

Car Information : BMW, 2019
Car owner : malpani
Miles Drive : 120
Original Car Price : 400000.0
Current Car Price : 399800.0
-----

Object of class is Created and Init is Called.
Object of class is Created and Init is Called.
-----

Car Information : Audi, 2017
Car owner : vikas
Miles Drive : 50
Original Car Price : 1080000.0
Current Car Price : 1079000.0
-----
```

**1\_11. Write about Operator Overloading. Perform Matrix Addition, Subtraction & Multiplication using Class Matrix & operator overloading. Overload toString() function in Matrix class.**

**CODE:**

```
fun main(){    val rows = 3    val columns = 2    val firstMatrix
= arrayOf(intArrayOf(6, 3), intArrayOf(9,0), intArrayOf(5, 4))
val secondMatrix = arrayOf(intArrayOf(2, 3), intArrayOf(9, 0),
intArrayOf(0, 4))    println("Matrix1: (3 * 2 Matrix): ")    for
(row in firstMatrix) {        for (column in row) {
print("$column ")
}
println(
)
}
```

```

        println("Matrix2: (3
* 2 Matrix): ")    for
(row in secondMatrix) {
    for (column in row) {
        print("$column ")

    }
    println()
}

    val sum = Array(rows) { IntArray(columns)
}    for (i in 0..rows - 1) {        for (j
in 0..columns - 1) {            sum[i][j] =
firstMatrix[i][j] + secondMatrix[i][j]
        }
    }

    println("Addition: (3
* 2 Matrix): ")    for
(row in sum) {        for
(column in row) {
        print("$column ")

    }
    println()
}

    val sab = Array(rows) {
IntArray(columns) }    for (i
in 0..rows - 1) {        for (j
in 0..columns - 1) {
        sab[i][j] = firstMatrix[i][j] - secondMatrix[i][j]
    }
}

    println("Subtraction: (3
* 2 Matrix): ")    for (row
in sab) {        for (column
in row) {
        print("$column ")

    }
    println()
}

v
a
l
r
1
=
2

```

```

V
a
l
c
1
=
3
V
a
l
r
2
=
3

V
a
l
c
2
=
2

    val firstMatrixx = arrayOf(intArrayOf(3, -2, 5), intArrayOf(3, 0,
4))    val secondMatrixx = arrayOf(intArrayOf(2, 3), intArrayOf(-9,
0), intArrayOf(0, 4))    val product =
multiplyMatrices(firstMatrixx, secondMatrixx, r1, c1, c2)

    displayProduct(product)
}

fun displayProduct(product:
Array <IntArray>) {
println("Product of two
matrices is : ")    for (row in
product) {        for (column
in row) {
print("$column ")
}
println(
)
    }
}

fun multiplyMatrices(firstMatrix:
Array <IntArray>,
secondMatrix: Array <IntArray>,
r1: Int,                c1:
Int,
                c2: Int):
Array <IntArray> {    val product
= Array(r1) { IntArray(c2) }
for (i in 0..r1 - 1) {        for
(j in 0..c2 - 1) {            for
(k in 0..c1 - 1) {

```



```
                product[i][j] += firstMatrix[i][k] *  
secondMatrix[k][j]  
            }  
  
        }  
    }  
    return product  
}
```

**OUTPUT:**

```
Connected to the target VM, address: '127.0.0.1:55949', transport: 'socket'  
Matrix1: (3 * 2 Matrix):  
6 3  
9 0  
5 4  
Matrix2: (3 * 2 Matrix):  
2 3  
9 0  
0 4  
Addition: (3 * 2 Matrix):  
8 6  
18 0  
5 8  
Subtraction: (3 * 2 Matrix):  
4 0  
0 0  
5 0  
Product of two matrices is :  
24 29  
6 25  
Disconnected from the target VM, address: '127.0.0.1:55949', transport: 'socket'  
  
Process finished with exit code 0  
|
```







