**Go pointers**

Go tasks easily perform by the pointer

Some cases such as call by reference will not perform without pointer

Every variable has a memory location

Memory location has address and can be accesses by & which is the address of mlocation

A **Pointers are the variable whose value is the address of another memory location**

var var\_name \*var-type

package main

import "fmt"

func main() {

    var x int = 20

    var y \*int

    fmt.Println(y) // THis is nil pointer where we hav not allocated the adress just we initialized

    y = &x

    fmt.Println(&x)

    fmt.Println(y)

    fmt.Println(\*y)

}

<nil>

0xc000014088

0xc000014088

20

**Passing array to function**

**void myFunction(param [10]int)**

**{**

**.**

**.**

**.**

**}**

**void myFunction(param []int)**

**{**

**.**

**.**

**.**

**}**

If you want to pass a single-dimension array as an argument in a function, you would have to declare function formal parameter in one of following two ways and all two declaration methods produce similar results because each tells the compiler that an integer array is going to be received. Similar way you can pass multi-dimensional array as formal parameters.

package main

import "fmt"

func main() {

    a := []int{0, 100, 52, 30}

    x := average(a, 4)

    fmt.Println(x)

}

func average(y []int, size int) float64 {

    var b float64

    var sum int

    for i := 0; i < size; i++ {

        sum += y[i]

    }

    b = (float64)(sum / size)

    return b

}

45

**Passing pointer to function**

**package main**

**import "fmt"**

**func main() {**

**var x int = 100**

**var y int = 200**

**fmt.Println(x, y)**

**swap(&x, &y)**

**fmt.Println(x, y)**

**}**

**func swap(a \*int, b \*int) int {**

**var tmp int**

**tmp = \*b**

**\*b = \*a**

**\*a = tmp**

**return 0**

**}**

**100 200**

**200 100**