**Exercise#1: Find the max element in an array**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **23** | **65** | **12** | **87** | **12** | **343** | **098** | **12** | **76** | **312** |

**Algorithm**

* **Pre-condition:**

To set the values for in array to find a maximum number. in this case array variable should be integer. write a code for a second variables(maxnumber) to find array numbers zero position. var1[0].

* **Loop variant**

In the for-loop condition i=1 is a loop variant. i is variables it is used to compare with max number. After find the max number next condition will be move on.

* **Loop invariant**

In the for-loop condition total length of the number is invariant condition. That will not be changed.

* **Post-Condition**

After iterating through the entire array, the maximum variable will hold the maximum number variables. Output should come max number.

**Pseudocode**

int maxnumber

{

int var1[array]={23,65,12,87,12,343,98,12,76,312}

int maxnumber =var1[0]

for (int i=1; i<number. Length; i++)

{

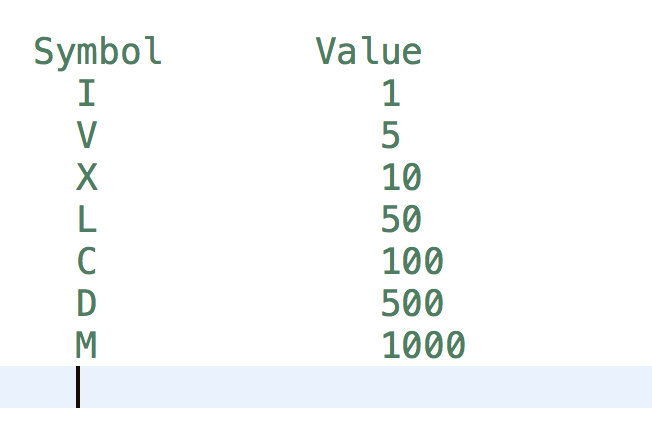
If number[array]>maxnumber {

maxnumber=number[array]

}

Print(maxnumber)

**Exercise#2: CONVERT ROMAN INTO INTEGER**



**Algorithm:**

**Precondition**: Before calling the roman\_to\_int () function, you must ensure that the input s meets the following conditions:

s is a non-empty string.

s contains only valid Roman numeral characters: 'I', 'V', 'X', 'L', 'C', 'D', and 'M'.

The Roman numerals in s are ordered according to the rules of Roman numerals, with larger numerals preceding smaller ones.

**Loop Invariant:** During the loop, the variable result keeps track of the sum of the decimal values of the Roman numerals processed so far. At the start of each iteration of the loop, result contains the sum of the decimal values of the Roman numerals processed up to the previous iteration.

**Loop Variant:** The loop processes one Roman numeral character at a time until all characters in the input string s have been processed. The loop variant is the number of characters remaining in s to be processed. At the start of each iteration of the loop, the loop variant decreases by 1, until it reaches 0 and the loop terminates.

**Postcondition**: After the roman\_to\_int() function is called with a valid input s, the function returns an integer value that represents the decimal equivalent of the input Roman numeral.

**Pseudocode:**

def roman\_to\_int(s):

roman\_dict = { 'I': 1, 'V': 5,'X': 10,'L': 50, 'C': 100,'D': 500,'M': 1000 }

result = 0

prev\_value = 0

for c in s:

value = roman\_dict[c]

if value > prev\_value:

result += value - 2 \* prev\_value

else:

result += value

prev\_value = value

   return result

numeral = 'XIV'

integer = roman\_to\_int(numeral)

print(integer)

**Exercise#3: Move zeroes to the end of Array**

* **Pre-condition:**

This case array variable should be integer.  if the current element is non-zero(i!=0), put the element at the next free position in the array.

* **Loop variant**

In the while loop condition i=0 i! =0 is a loop variant.

* **Loop invariant**

in the while-loop condition total number of the length is in variant condition.

* **Post-Condition**

If the current element i=0 move 0’s to the end of the array.

**Pseudocode**

int i {

while (i! = 0) {

i = i – 1

}

int i = 0

while (i=0) {

i = i + 1

print(i)

} Here 1 and -1 is use to move position of array

}