M.S. Ramaiah Institute of Technology
(Autonomous Institute, Affiliated to VTU)
Department of Computer Science and Engineering

Module 3

UNIT 3

Arrays and Matrices: One-Dimensional Array, Two-Dimensional Arrays (Declaration and Compile Time and Run Time Initialization). Sorting Algorithms, Search Algorithms
Character Arrays and Strings: Introduction, Declaring and Initializing String Variables, Reading Strings from Terminal, Writing Strings to Screen, Arithmetic Operations on Characters, String Handling Functions, Other String Functions.

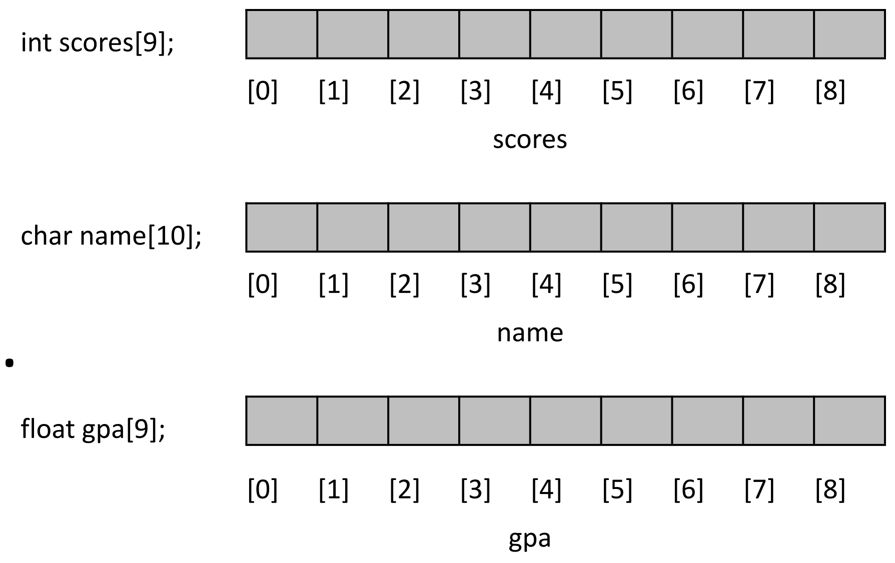
Why Arrays?

- Suppose, you need to store years of 100 cars. Will you define 100 variables?
  – int y1, y2,…, y100;

- *Use the data structure Array:* An array is an indexed data structure to represent several variables having the same data type:

Arrays

- Array is a data structure which collects related data items that share a common array name and data type is called an array.
- **Example: marks[3]**
  - Represents the marks of the 7 students.
  - The individual values are called elements.
    - Marks[0],marks[1],marks[2]
    - are the individual elements of the array.



Properties of an Array

- An array’s data items are stored contiguously in memory.
- An array is a group of homogeneous data items that all have the same array name and same data type.
- Arrays are static in that they remain the same size throughout program execution.
- Each element can be accessed individually using index or subscript

One-dimensional array

- When a listof items can be given one variable name using only one subscript and such avariable is called a one dimensional array.
- int num[5];

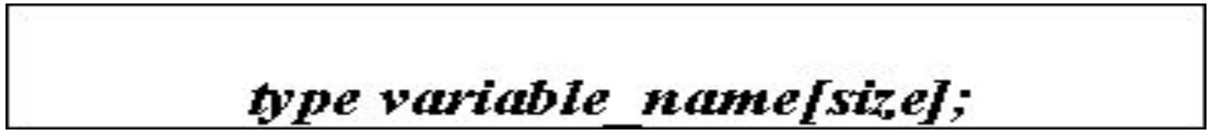
Num[0]
Num[1]
Num[2]
Num[3]
Num[4]

- The subscript of an array always start with 0.

Declaration and Initialization of Arrays

Declaration:

The arrays are declaredbefore they are used in the program. The general form of array declaration is

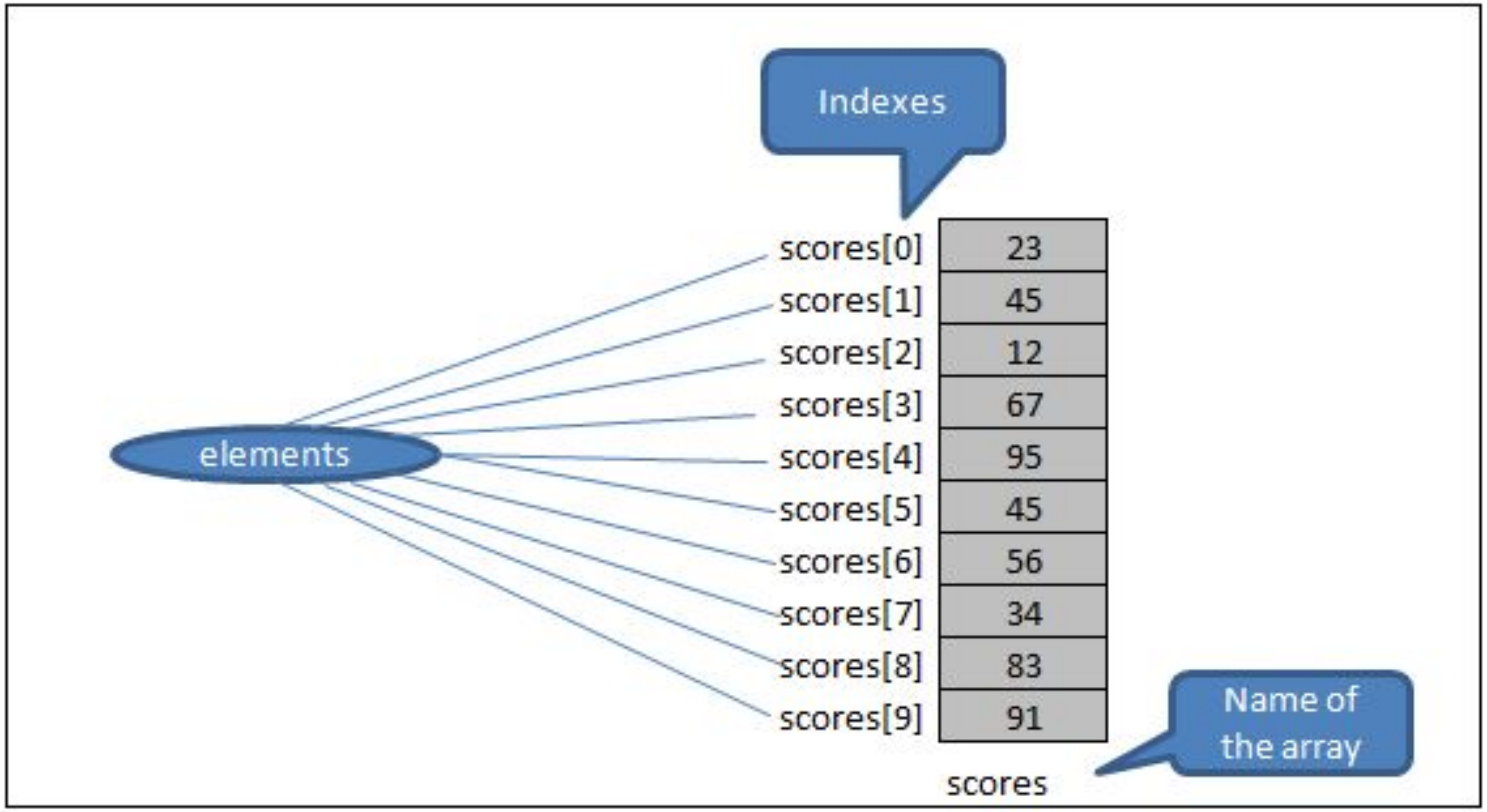


```
type variable_name[size];
```

The *type*specifies the type of element that will be contained in the array, such as int, float,or char.

size indicates the maximum number of elements that can be stored inside the array.

Using arrays in c



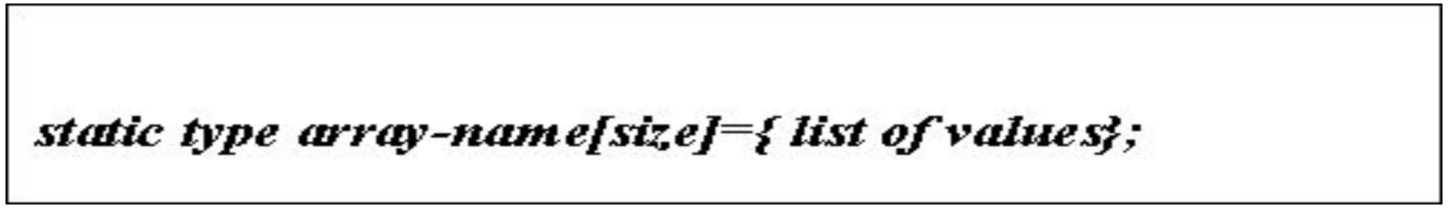
The scores Array

Declaration and Initialization of Arrays

- **Example:**
- **1. float weight[40];**
  - Declares the weight to be an array containing 40 real elements. Any subscripts 0 to 39 are valid.
- **2. int group1[11];**
  - Declares the group1 as an array to contain a maximum of 10 integer constants.

Initialization of arrays

- The general form of initialization of arrays is:



```
static type array-name[size]={ list of values};
```

- The values in the list are separated by commas.
- For example, the statement below shows
  - *int num[3]={2,2,2};*

Initialization of arrays

- Points to remember:
  - If the number of values is less than the number of that many elements will be initialized. elements, then only
  - The remaining elements will be set to zero automatically.
- float num1[5]={0.1,2.3,4.5};
- The first three elements are initialized to 0.1,2.3 and 4.5 and the remaining two elements to zero.

Compile Time Initialization

- Arrays with the size
  - int num[3]={1,2,3}
- Arrays without size

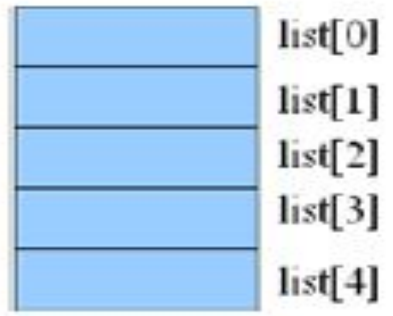
– int count[ ]= {2,2,2,2};

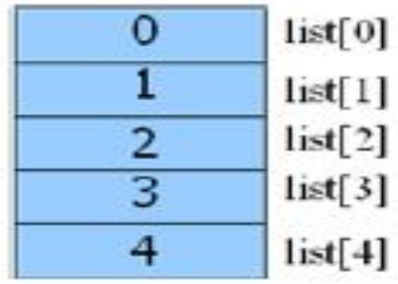
- Character arrays may be initialized in a similar manner.

– char name[ ]={‘S’,’W’,’A’,’N’}

Run time Initialization

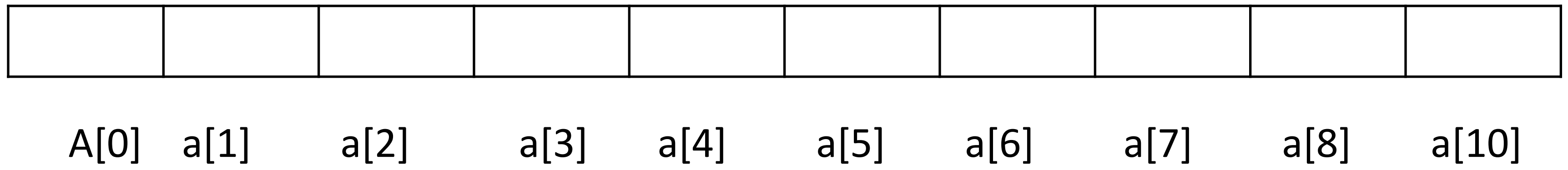
```
int list[5], i;
for(i=0; i<5; i++){
            list[i] = i;
}
OR
for(i=0; i<=4; i++){
   list[i] = i;
}
```





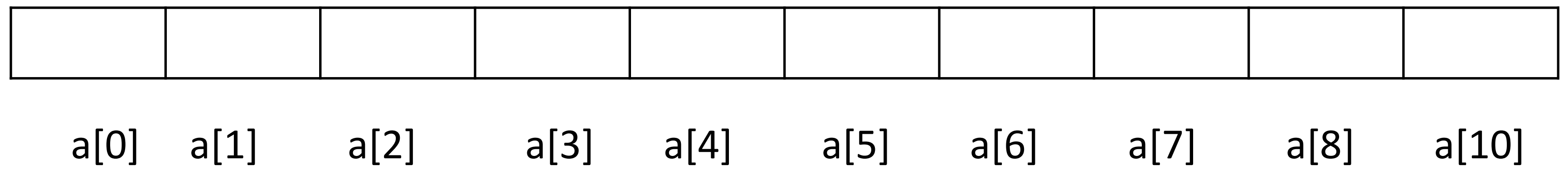
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)
{
        scanf(“%d”,&a[i]);
}
```



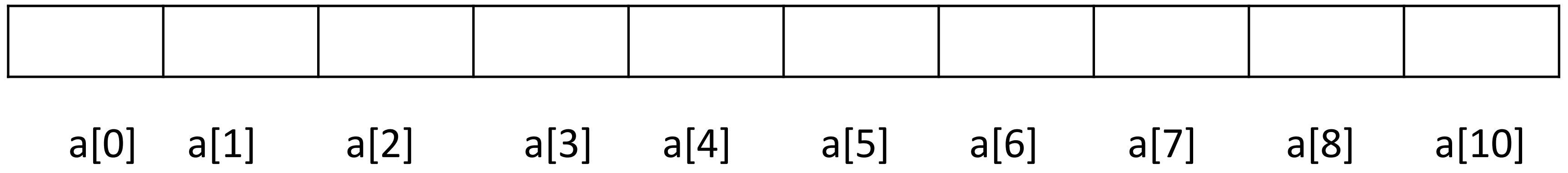
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)
{
        scanf("%d",&a[i]);
}
```



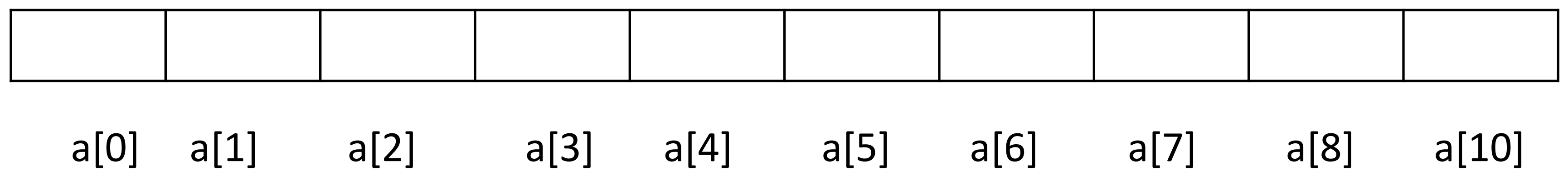
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)
{
        scanf("%d",&a[i]);
}
```



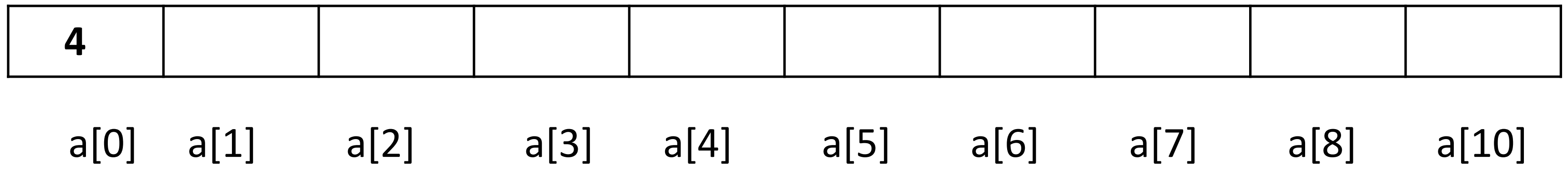
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)  // 0<=4
{
        scanf("%d",&a[i]);
}
```



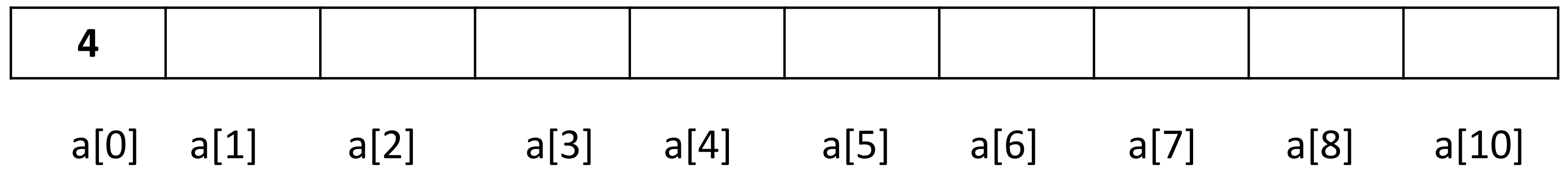
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)
{
        scanf(“%d”,&a[i]);  // a[0]
}
```



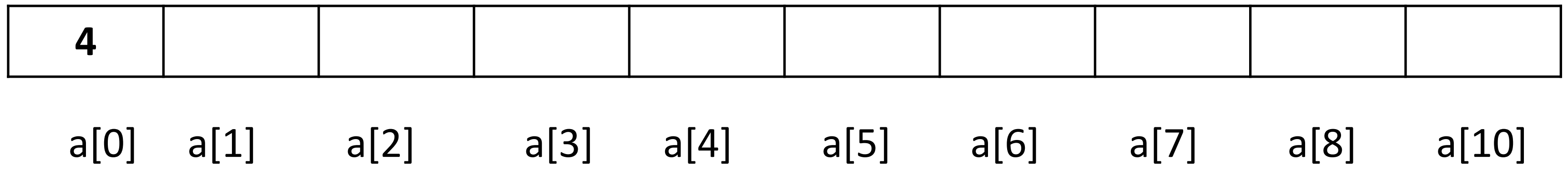
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++) \\ increment i=1
{
        scanf("%d",&a[i]);
}
```



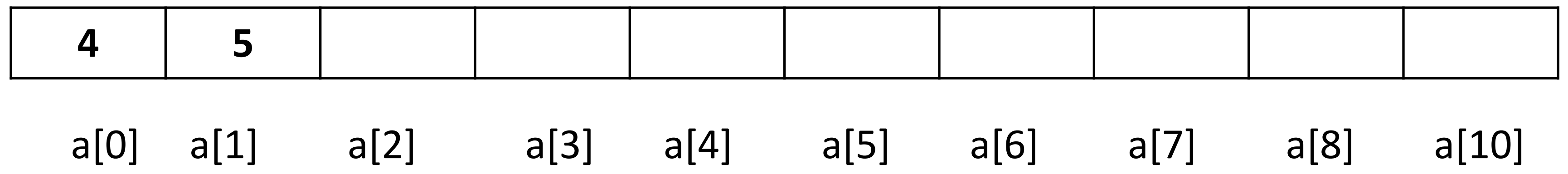
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)   1<=4
{
        scanf("%d",&a[i]);
}
```



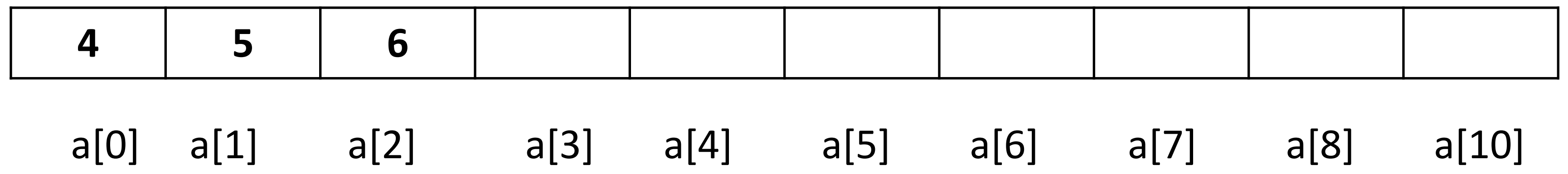
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)
{
        scanf(“%d”,&a[i]);     a[1]
}
```



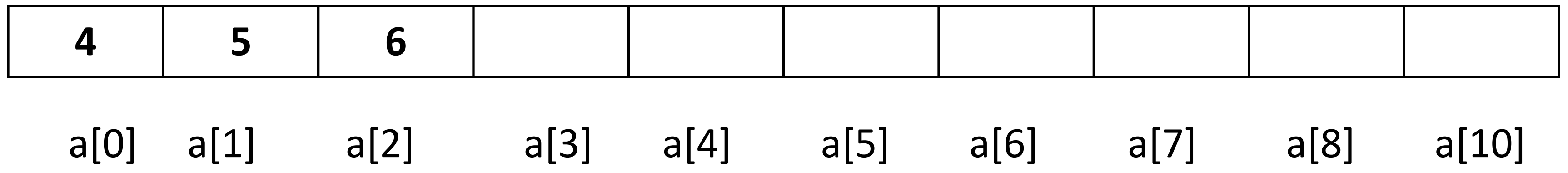
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)     3<=4
{
        scanf("%d",&a[i]);
}
```



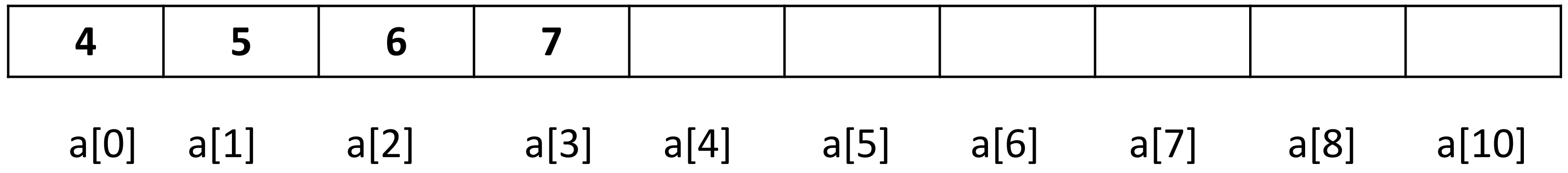
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)
{
        scanf("%d",&a[i]);
}
```



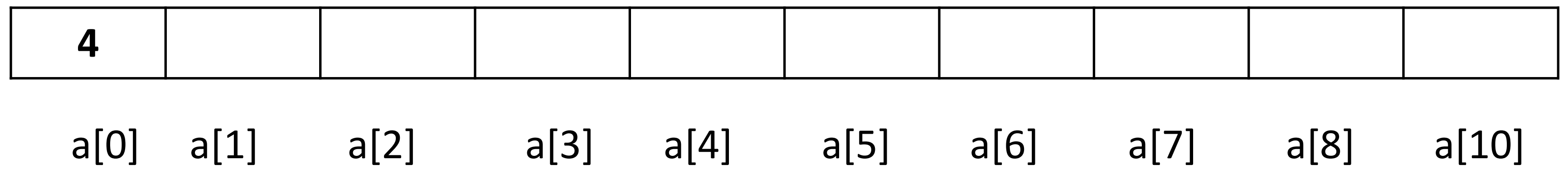
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)
{
        scanf("%d",&a[i]);
}
```



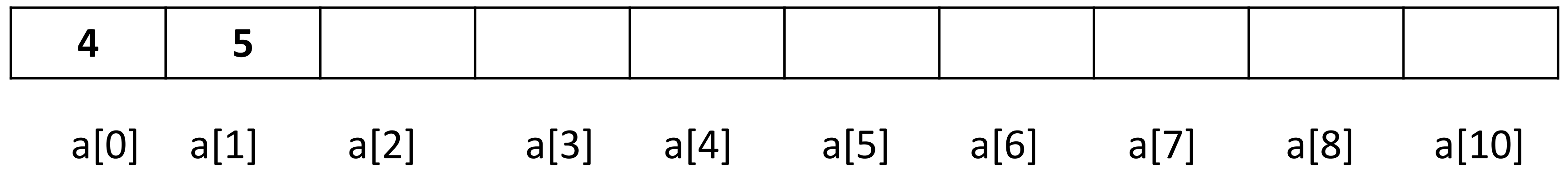
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)   1<=4
{
        scanf("%d",&a[i]);
}
```



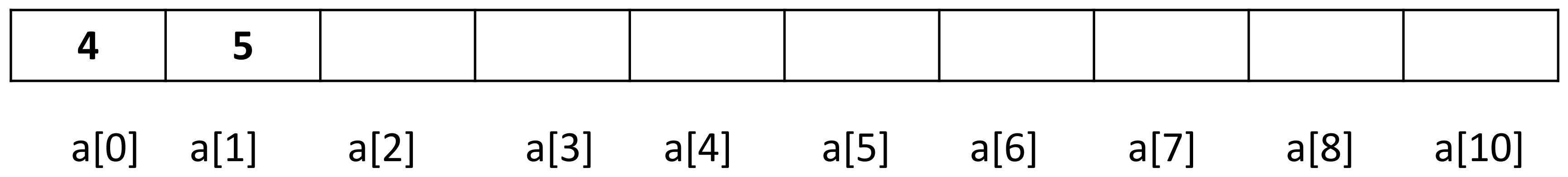
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)
{
        scanf(“%d”,&a[i]);     a[1]
}
```



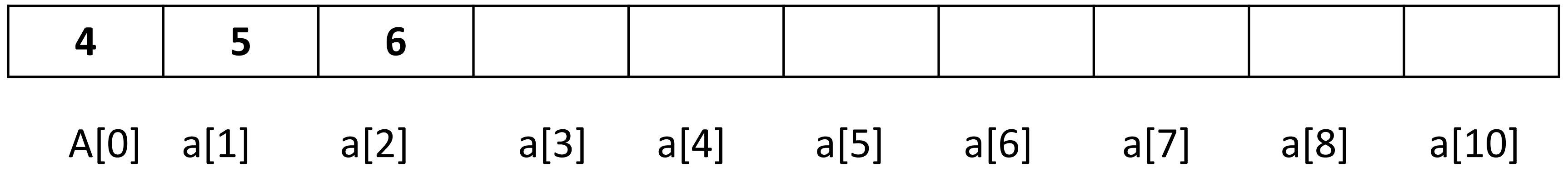
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)  i=2
{
        scanf("%d",&a[i]);
}
```



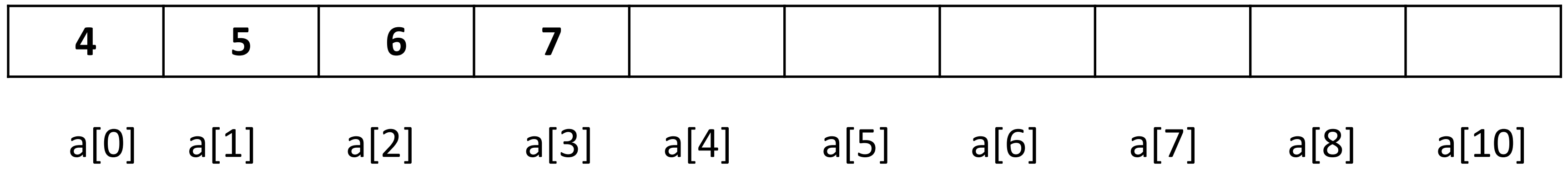
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)      2<=4
{
        scanf("%d",&a[i]);
}
```



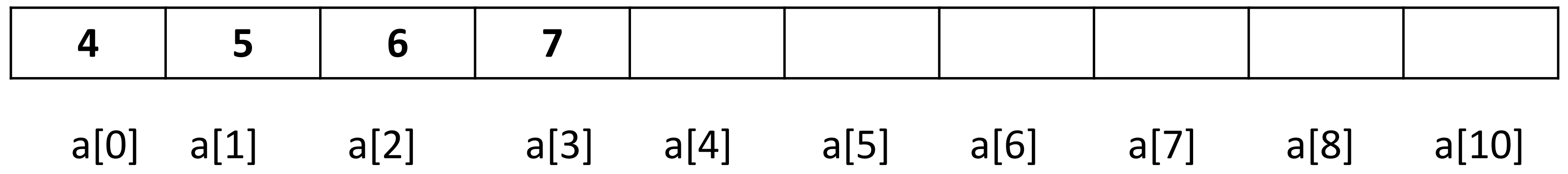
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)   i=4
{
        scanf("%d",&a[i]);
}
```



Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)   4<=4
{
        scanf("%d",&a[i]);
}
```



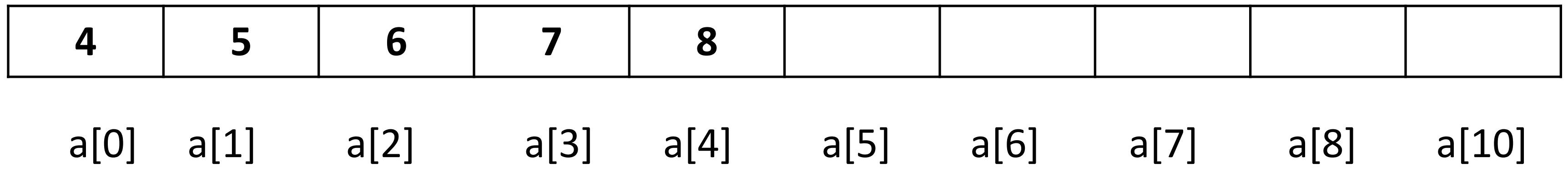
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)
{
        scanf("%d",&a[i]);    a[4]
}
```

4	**5**	**6**	**7**	**8**					
a[0]	a[1]	a[2]	a[3]	a[4]	a[5]	a[6]	a[7]	a[8]	a[10]

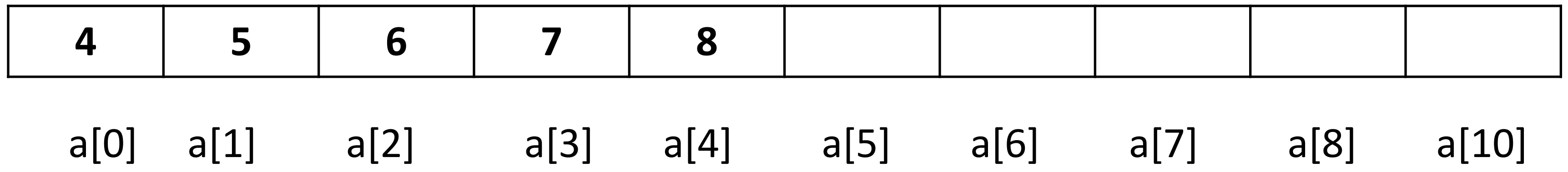
Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)   i=5
{
        scanf("%d",&a[i]);
}
```



Read elements into array

```
Int a[10];
for(i=0;  i<=4; i++)   5<=4 false
{
        scanf("%d",&a[i]);
}
```



Simple Array Programs

```
// Program to take 5 values from the user and store them in an array // Print the
elements stored in the array
 #include <stdio.h>
 int main()
 {
 int values[5];
 printf("Enter 5 integers: "); // taking input and storing it in an array
 for(int i = 0; i < 5; ++i)
 {
scanf("%d", &values[i]);
 }
 printf("Displaying integers: "); // printing elements of an array
for(int i = 0; i < 5; ++i)
{
 printf("%d\n", values[i]);
 } return 0; }
```

Write a program to calculate average of the numbers in an array

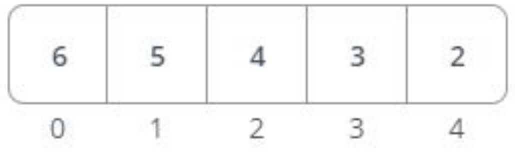
```
int main()
{
        int array[10] = {1, 2, 3, 4, 5, 6, 7, 8, 9, 0};
        int sum, loop;
        float avg;
        sum = avg = 0;
        for(loop = 0; loop < 10; loop++)
        {
                sum = sum + array[loop];
        }
        avg = (float)sum / loop;
        printf("Average of array values is %.2f", avg);
        return 0;
}
```

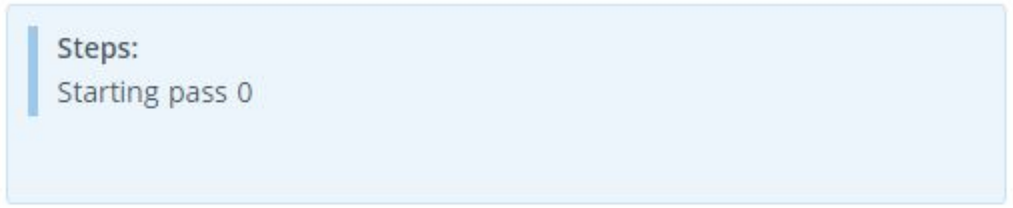
Write a program to sort the list using bubblesort

```
#include <stdio.h>
intmain()
{
   intarray[100], n, i, j, swap;
   printf("Enter number of elements\n");
   scanf("%d", &n);
   printf("Enter %d integers\n", n);
   for (i = 0; i< n; i++)
         scanf("%d", &array[i]);
```

```
Output:
Enter number of elements: 5
Enter 5 integers 4       5   3     2     1
```

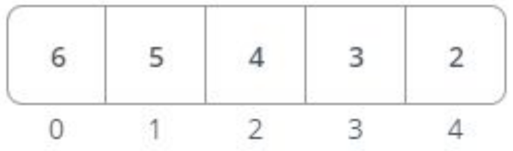
```
for (i = 0 ; i < n - 1; i++)
 {
   for (j = 0 ;j< n - i - 1; j++)
   {
          if (array[j] > array[j+1])
{
               swap      = array[j];
               array[j]   = array[j+1];
               array[j+1] = swap;
          }
    }
 }

printf("Sorted list in ascending order:\n");
  for (i = 0; i< n; i++)
```





Steps:
Starting pass 0

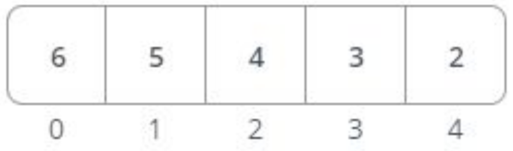
```
for (i = 0 ; i < n - 1; i++)
 {
       for (j = 0 ;j< n - i - 1; j++)
       {
              if (array[j] > array[j+1])
{
                    swap      = array[j];
                    array[j]   = array[j+1];
                    array[j+1] = swap;
              }
        }
 }
```

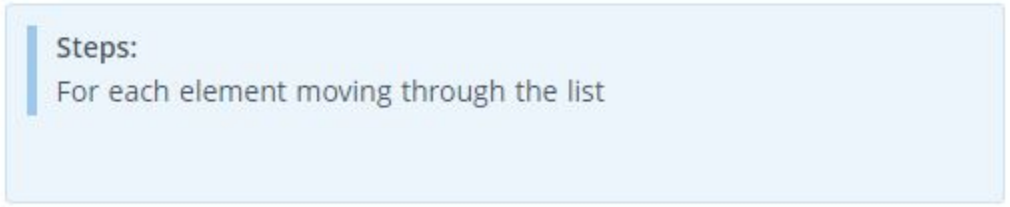


Steps:

For each element moving through the list

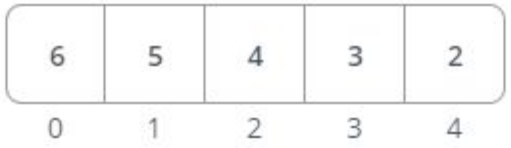
```
for (i = 0 ; i < n - 1; i++)  i<4
 {
        for (j = 0 ;j< n - i - 1; j++)
        {
                if (array[j] > array[j+1])
{
                        swap       = array[j];
                        array[j]   = array[j+1];
                        array[j+1] = swap;
                }
        }
 }
```





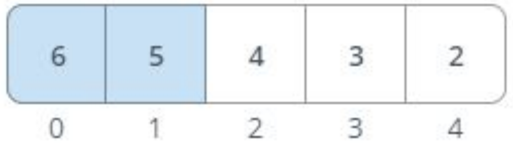
Steps:
For each element moving through the list

```
for (i = 0 ; i < n - 1; i++)  i<4
 {
      for (j = 0 ;j< n - i - 1; j++)
      {
            if (array[j] > array[j+1])
{
                 swap      = array[j];
                 array[j]   = array[j+1];
                 array[j+1] = swap;
            }
      }
 }
```



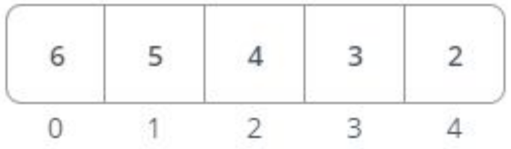
Steps:
For each element moving through the list

```
for (i = 0 ; i < n - 1; i++)  i<4
 {
       for (j = 0 ;j< n - i - 1; j++) 0< 5-0-1
       {
              if (array[j] > array[j+1])
{
                    swap       = array[j];
                    array[j]   = array[j+1];
                    array[j+1] = swap;
              }
        }
 }
```



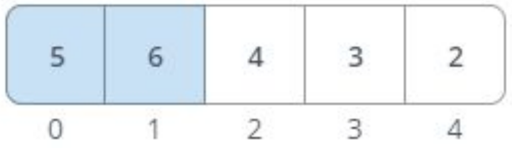
Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  i<4
  {
        for (j = 0 ;j< n - i - 1; j++)
        {                              6>5
                if (array[j] > array[j+1])   array[0]>array[1]
{
                      swap      = array[j];
                      array[j]   = array[j+1];
                      array[j+1] = swap;
                }
         }
  }
```

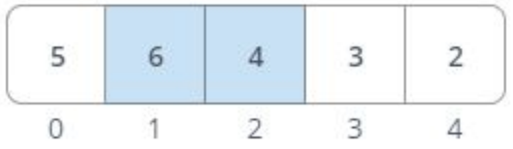


Steps:
For each element moving through the list

```
for (i = 0 ; i < n - 1; i++)  i<4
  {
        for (j = 0 ;j< n - i - 1; j++)
        {
               if (array[j] > array[j+1])   array[0]>array[1]
{
                     swap      = array[j];   swap=array[0]  ie 6
                     array[j]   = array[j+1]; array[0]= array[1]  5
                     array[j+1] = swap;  array[1]=6
               }
         }
  }
```

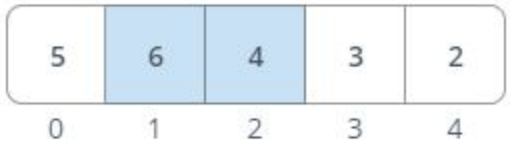


Steps:
Swap



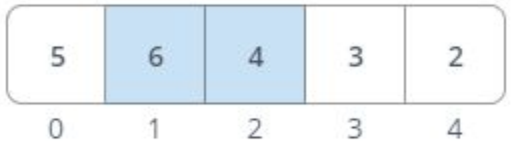
Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  i<4
 {
       for (j = 0 ;j< n - i - 1; j++)
       {
              if (array[j] > array[j+1])
              {
                     swap      = array[j];
                     array[j]   = array[j+1];
                     array[j+1] = swap;
              }
       }
 }
```



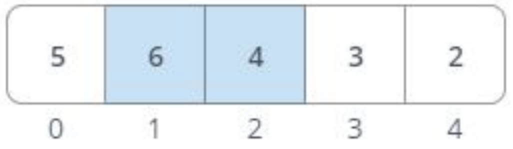
Steps:
Compare elements

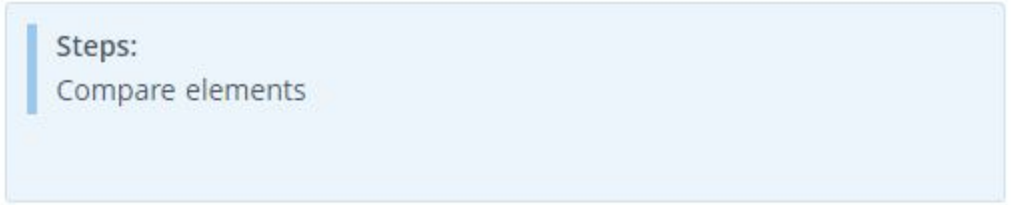
```
for (i = 0 ; i < n - 1; i++)  i<4
 {
      for (j = 0 ;j< n - i - 1; j++) 1<5-0-1
      {
             if (array[j] > array[j+1])
             {
                    swap       = array[j];
                    array[j]   = array[j+1];
                    array[j+1] = swap;
             }
       }
 }
```



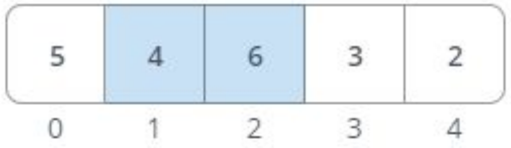
Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  i<4
 {
      for (j = 0 ;j< n - i - 1; j++)
      {                                6>4
            if (array[j] > array[j+1])    array[1]>array[2]
            {
                  swap       = array[j];
                  array[j]   = array[j+1];
                  array[j+1] = swap;
            }
      }
 }
```

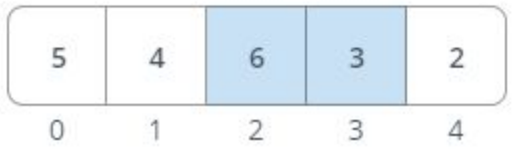



Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  i<4
 {
       for (j = 0 ;j< n - i - 1; j++)
       {
               if (array[j] > array[j+1])     6>4
{
                      swap      = array[j];  swap=6
                      array[j]   = array[j+1];  array[1] =4
                      array[j+1] = swap;  array[2]=6
               }
         }
 }
```

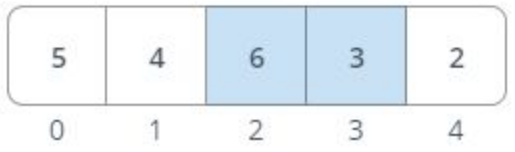


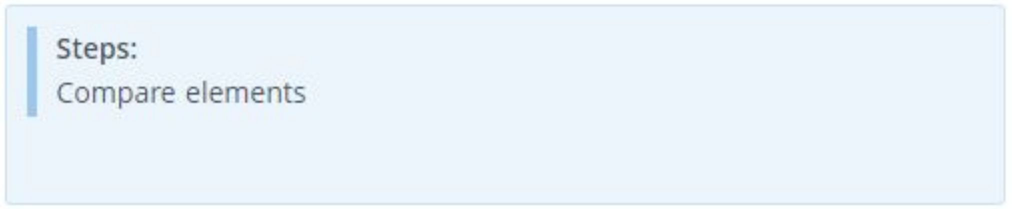
Steps:
Swap



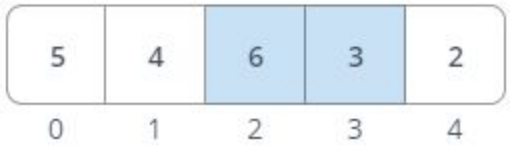
Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  i<4
 {
        for (j = 0 ;j< n - i - 1; j++) J=2
{
                if (array[j] > array[j+1])
               {
                    swap       = array[j];
                    array[j]   = array[j+1];
                    array[j+1] = swap;
               }
        }
 }
```



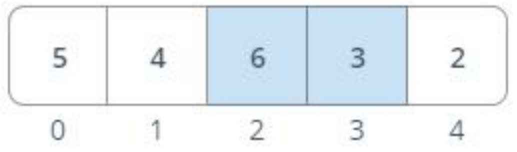

Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  i<4
 {
        for (j = 0 ;j< n - i - 1; j++) 2<5-0-1
{
               if (array[j] > array[j+1])
              {
                     swap      = array[j];
                     array[j]   = array[j+1];
                     array[j+1] = swap;
              }
        }
 }
```



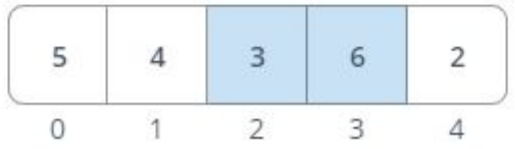
Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  i<4
 {
        for (j = 0 ;j< n - i - 1; j++) 2<5-0-1
{
                if (array[j] > array[j+1])      array[2]>array[3]
                {
                        swap       = array[j];
                        array[j]   = array[j+1];
                        array[j+1] = swap;
                }
        }
 }
```

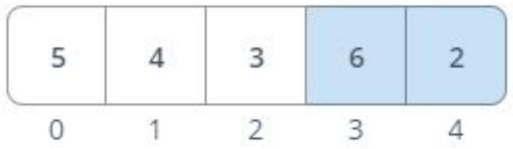


Steps:
Compare elements

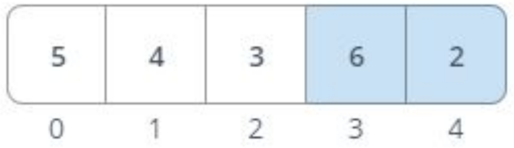
```
for (i = 0 ; i < n - 1; i++)  i<4
 {
        for (j = 0 ;j< n - i - 1; j++) 2<5-0-1
{
                if (array[j] > array[j+1])      array[2]>array[3]
                {
                        swap      = array[j];  =array[2]  6
                        array[j]   = array[j+1];  =array[3] 3
                        array[j+1] = swap;  =6
                }
        }
 }
```



Steps:
Swap

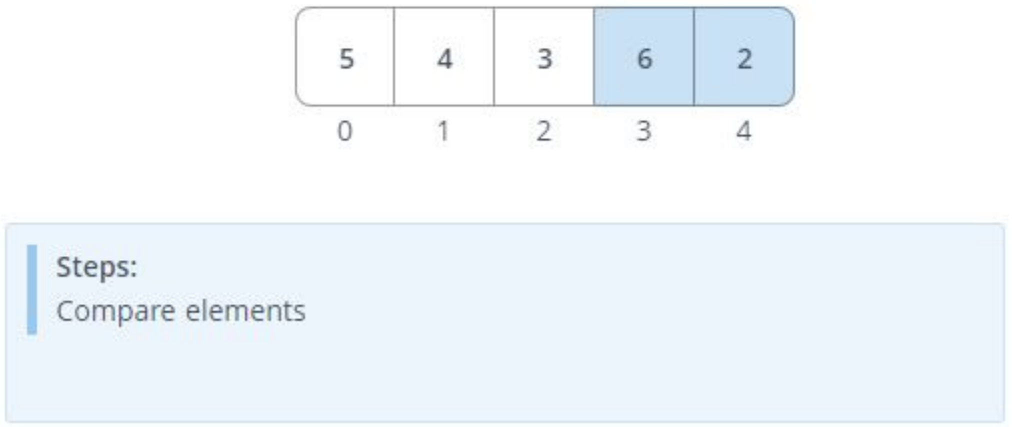


Steps:
Compare elements



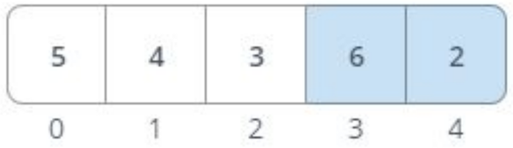
Steps:
Compare elements

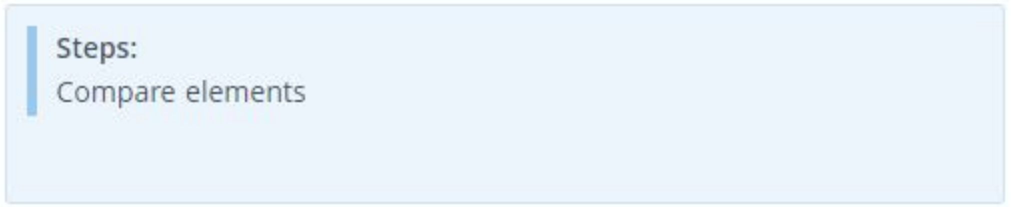
```
for (i = 0 ; i < n - 1; i++)  i<4

        or (j = 0 ;j< n - i - 1; j++)  3

                if (array[j] > array[j+1])
                {
                        swap       = array[j];
                        array[j]   = array[j+1];
                        array[j+1] = swap;
                }
        }
}
```



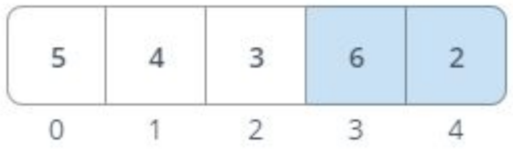
Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  i<4
 {
     for (j = 0 ;j< n - i - 1; j++) 3<4  5-0-1

          if (array[j] > array[j+1])
          {
               swap       = array[j];
               array[j]   = array[j+1];
               array[j+1] = swap;
          }
     }
}
```



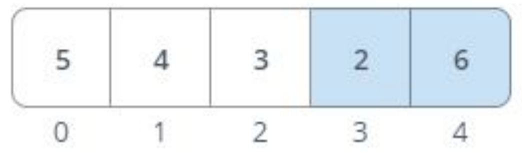

Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  i<4

        or (j = 0 ;j< n - i - 1; j++) 3<4  5-0-1
                                6>2
              if (array[j] > array[j+1])  array[3]>array[4]
              {
                     swap      = array[j];
                     array[j]   = array[j+1];
                     array[j+1] = swap;
              }
        }
}
```

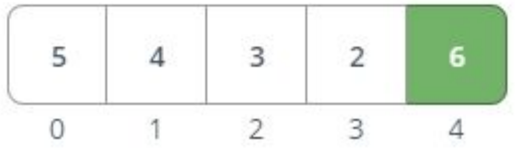


Steps:
Compare elements

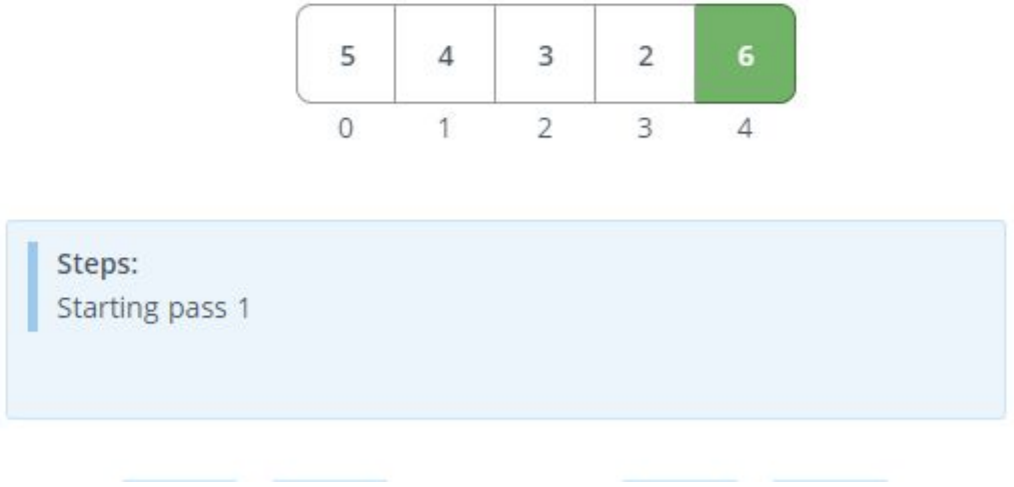
```
for (i = 0 ; i < n - 1; i++)  i<4

        or (j = 0 ;j< n - i - 1; j++) 3<4  5-0-1
                                 6>2
              if (array[j] > array[j+1])  array[3]>array[4]
              {
                    swap       = array[j];  =array[3]  6
                    array[j]   = array[j+1];  =array[4]  2
                    array[j+1] = swap; =6
              }
        }
}
```



Steps:
Swap

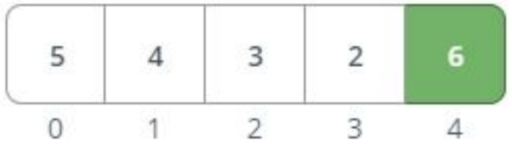


Steps:
Done this pass. The last element processed is now in its final position.



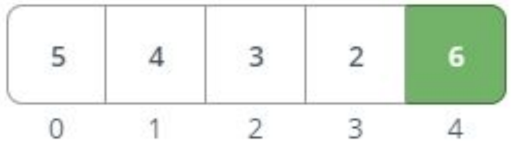
Steps:
Starting pass 1

```
for (i = 0 ; i < n - 1; i++)  i=1
 {
       for (j = 0 ;j< n - i - 1; j++)
{
               if (array[j] > array[j+1])
              {
                     swap       = array[j];
                     array[j]   = array[j+1];
                     array[j+1] = swap;
              }
        }
 }
```



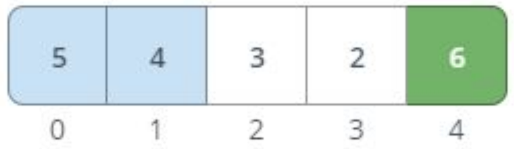
Steps:
For each element moving through the list

```
for (i = 0 ; i < n - 1; i++)  1<4
 {
        for (j = 0 ;j< n - i - 1; j++)
{
                 if (array[j] > array[j+1])
                {
                        swap      = array[j];
                        array[j]   = array[j+1];
                        array[j+1] = swap;
                }
        }
 }
```



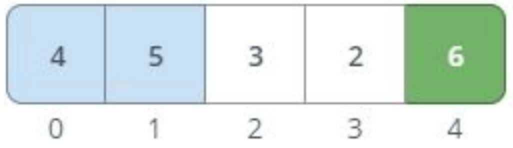
Steps:
For each element moving through the list

```
for (i = 0 ; i < n - 1; i++)  1<4
 {
        for (j = 0 ;j< n - i - 1; j++) 0< 5-1-1  0<3
{
                 if (array[j] > array[j+1])
                {
                        swap      = array[j];
                        array[j]   = array[j+1];
                        array[j+1] = swap;
                }
        }
 }
```

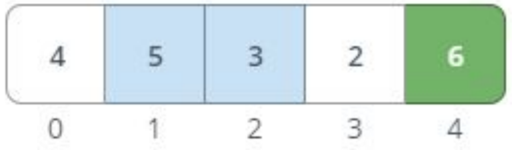


Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  1<4
 {
        for (j = 0 ;j< n - i - 1; j++) 0< 5-1-1  0<3
{
                 if (array[j] > array[j+1])      array[0]>array[1]
                {
                        swap       = array[j];
                        array[j]   = array[j+1];
                        array[j+1] = swap;
                }
          }
 }
```

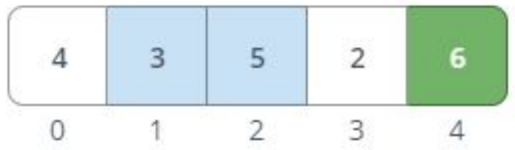


Steps:
Swap

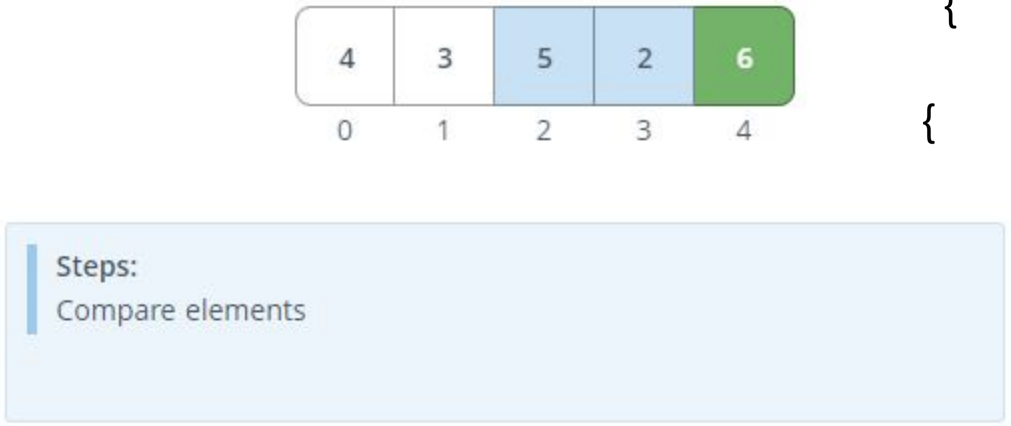


Steps:
Compare elements

```
for (i = 0 ; i < n - 1; i++)  1<4
  {                          1
        for (j = 0 ;j< n - i - 1; j++)   1< 5-1-1  1<3
{
                 if (array[j] > array[j+1])      array[1]>array[2]
                 {
                        swap       = array[j];
                        array[j]   = array[j+1];
                        array[j+1] = swap;
                 }
        }
  }
```

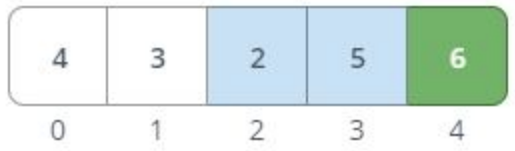


Steps:
Swap

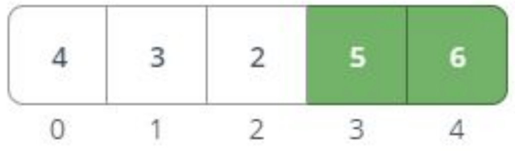


Steps:
Compare elements

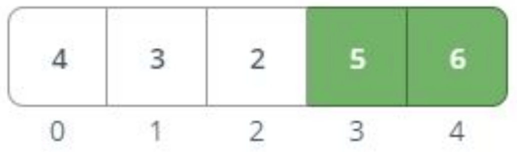
```
for (i = 0 ; i < n - 1; i++)  1<4
 {                        2
      for (j = 0 ;j< n - i - 1; j++) 2< 5-1-1  2<3
{
           if (array[j] > array[j+1]) array[2]>array[3]
           {
                swap       = array[j];
                array[j]   = array[j+1];
                array[j+1] = swap;
           }
      }
 }
```



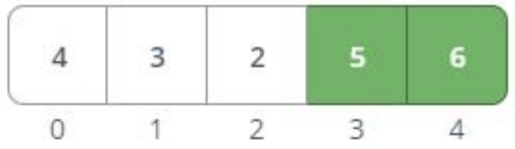
Steps:
Swap



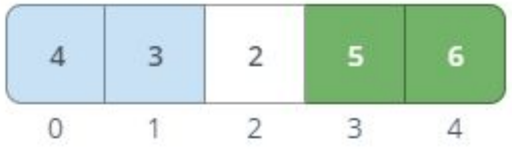
Steps:
Done this pass. The last element processed is now in its final position.



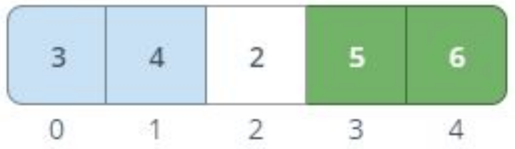
Steps:
Starting pass 2



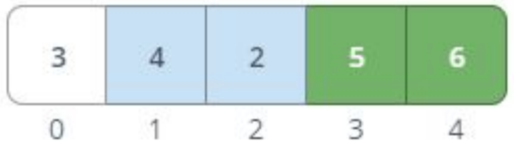
Steps:
For each element moving through the list



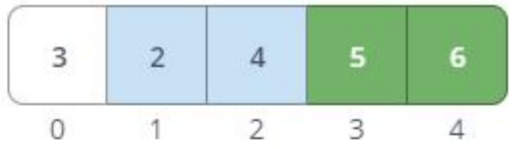
Steps:
Compare elements



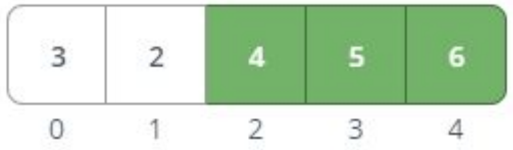
Steps:
Swap



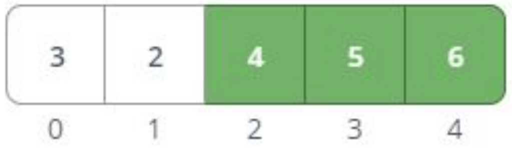
Steps:
Compare elements



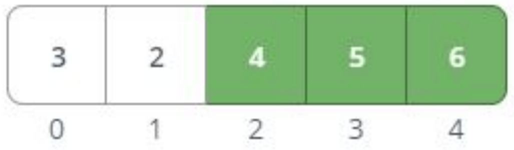
Steps:
Swap



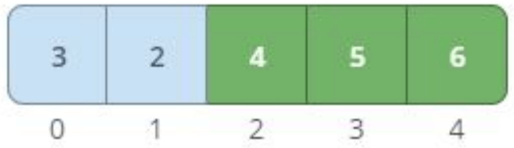
Steps:
Done this pass. The last element processed is now in its final position.



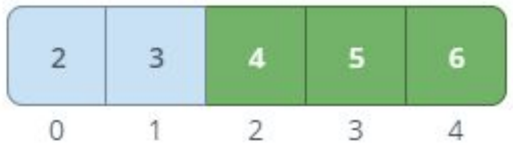
Steps:
Starting pass 3



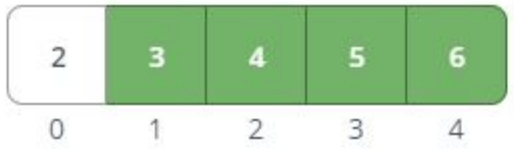
Steps:
For each element moving through the list



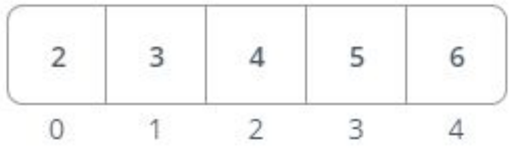
Steps:
Compare elements



Steps:
Swap



Steps:
Done this pass. The last element processed is now in its final position.



Steps:
Done sorting!

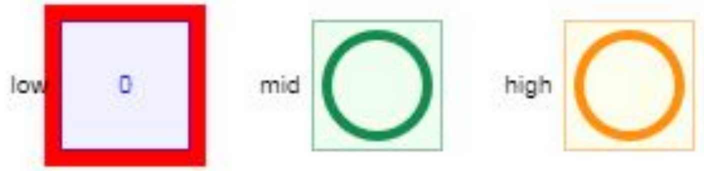
C program to search an ordered list using binary search

```
• #include <stdio.h>
• int main()
• {
•     int i, low, high, mid, n, key, array[100];
•     printf("Enter number of elementsn");
•     scanf("%d",&n);
•     printf("Enter %d integersn", n);
•     for(i = 0; i < n; i++)
•     scanf("%d",&array[i]);
•     printf("Enter value to findn");
•     scanf("%d", &key);
•     low = 0;
•     high = n - 1;
```

```
• while (low <= high)
•  {
•     mid = (low + high)/2;
•     if(array[mid] < key)
•             low = mid + 1;
•     else if (array[mid] == key)
•     {
•             printf("%d found at location %d.n", key, mid+1);
•             break;
•     }
•     else
•             high = mid - 1;
•     }
• if(low > high)
•     printf("Not found! %d isn't present in the list.n", key);
```

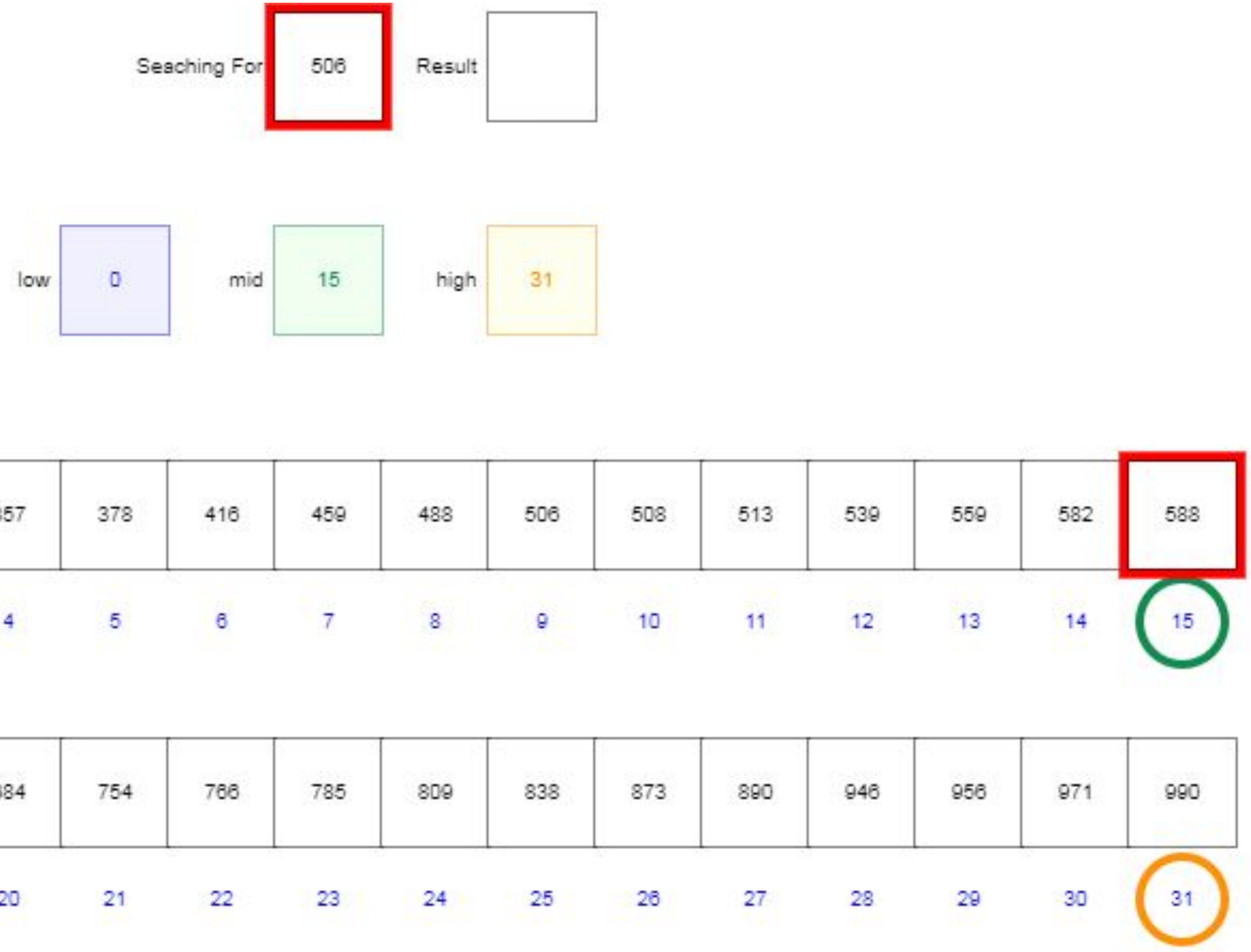
```
def binarySearch(listData, value)
  low = 0
  high = len(listData) - 1
  while (low <= high)
    mid = (low + high) / 2
    if (listData[mid] == value):
      return mid
    elif (listData[mid] < value)
      low = mid + 1
    else:
      high = mid - 1
  return -1
```

Seaching For	506	Result	



1	140	157	259	357	378	416	459	488	506	508	513	539	559	582	588
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
589	649	653	662	684	754	766	785	809	838	873	890	946	956	971	990
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

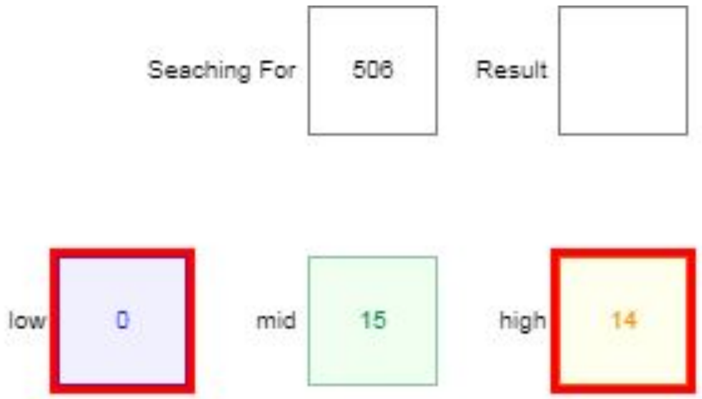
```
def binarySearch(listData, value)
    low = 0
    high = len(listData) - 1
    while (low <= high)
        mid = (low + high) / 2
        if (listData[mid] == value):
            return mid
        elif (listData[mid] < value)
            low = mid + 1
        else:
            high = mid - 1
    return -1
```

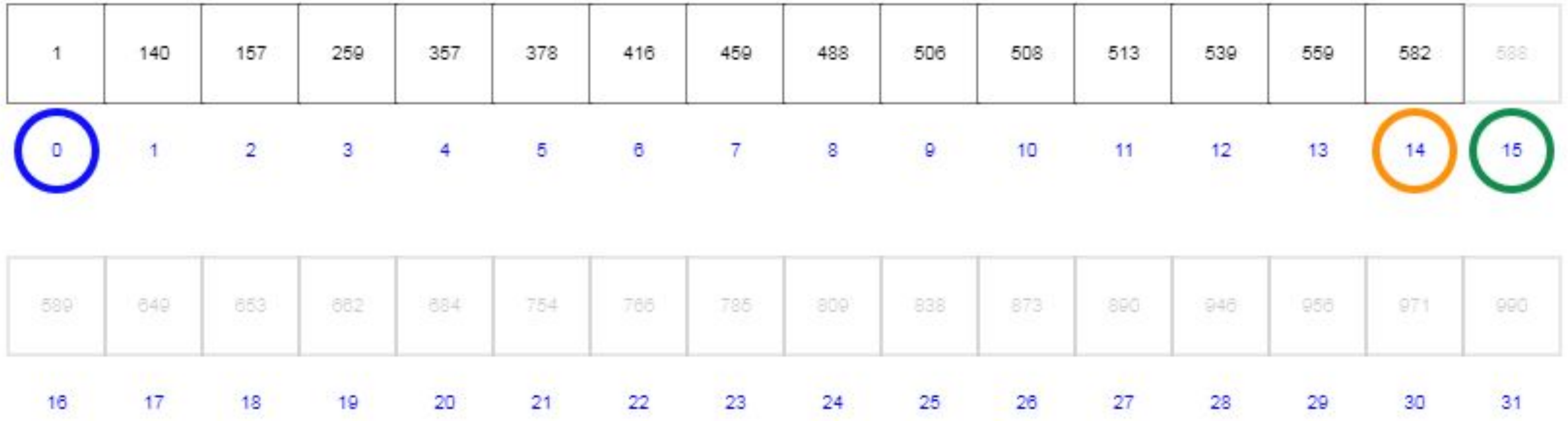


1	140	157	259	357	378	416	459	488	506	508	513	539	559	582	588
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

589	649	653	662	684	754	766	785	809	838	873	890	946	956	971	990
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

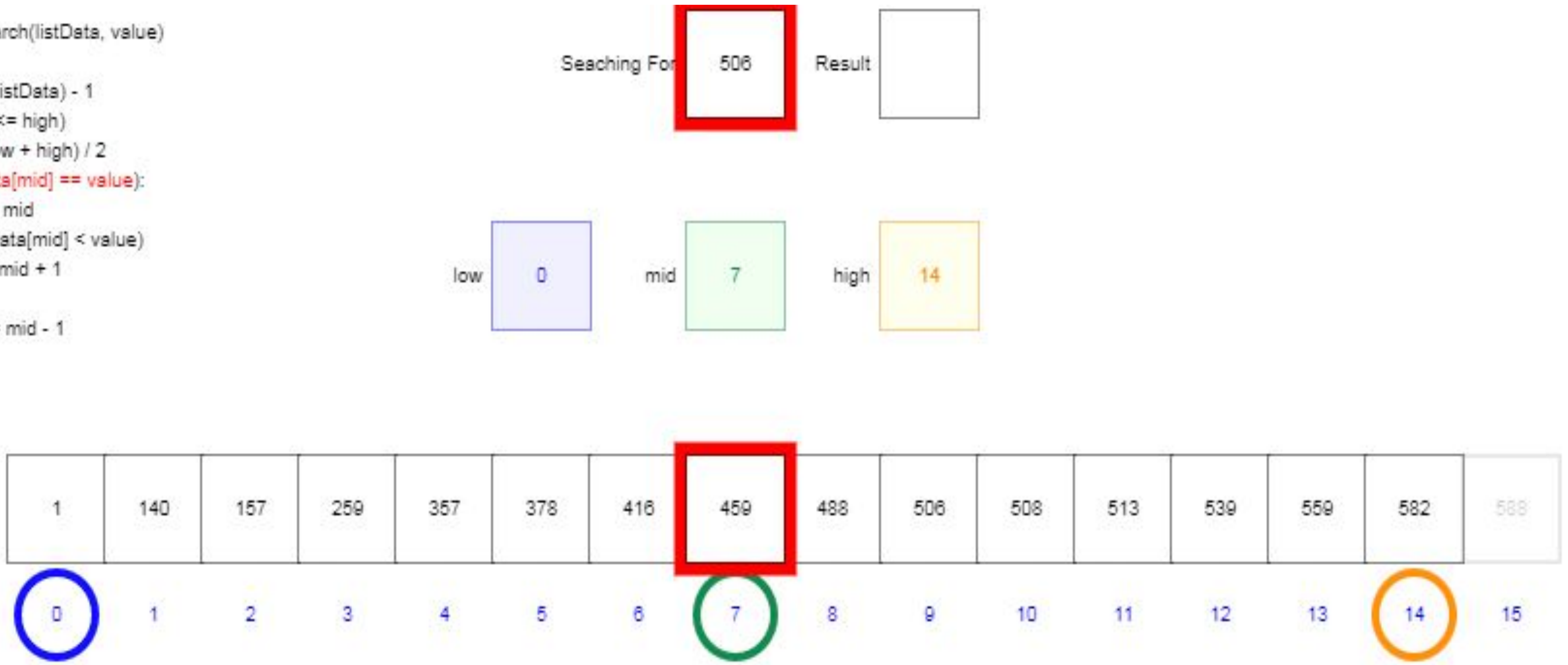
```
def binarySearch(listData, value)
  low = 0
  high = len(listData) - 1
  while (low <= high)
    mid = (low + high) / 2
    if (listData[mid] == value):
      return mid
    elif (listData[mid] < value)
      low = mid + 1
    else:
      high = mid - 1
  return -1
```



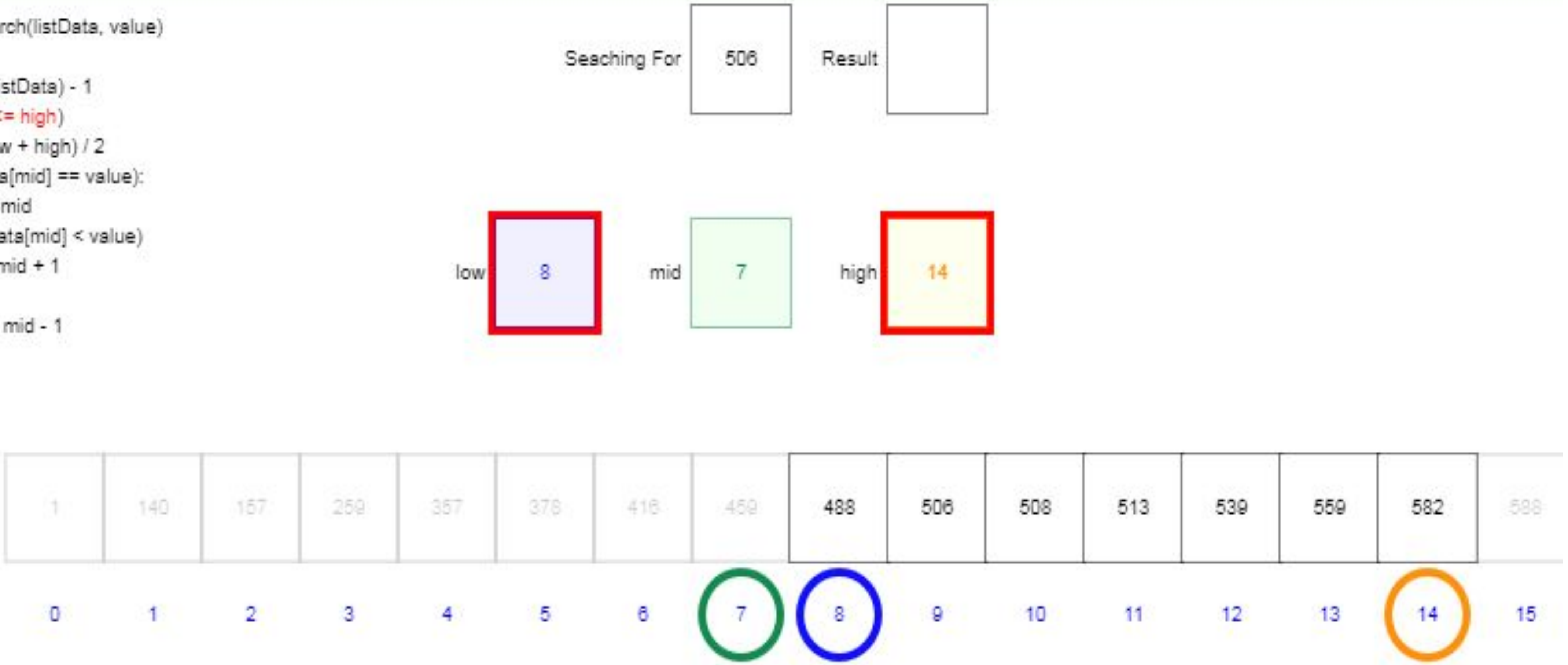


1	140	157	259	357	378	416	459	488	506	508	513	539	559	582	588
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
599	649	653	662	684	754	766	785	809	838	873	890	946	956	971	990
16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31

```
def binarySearch(listData, value)
    low = 0
    high = len(listData) - 1
    while (low <= high)
        mid = (low + high) / 2
        if (listData[mid] == value):
            return mid
        elif (listData[mid] < value)
            low = mid + 1
        else:
            high = mid - 1
    return -1
```



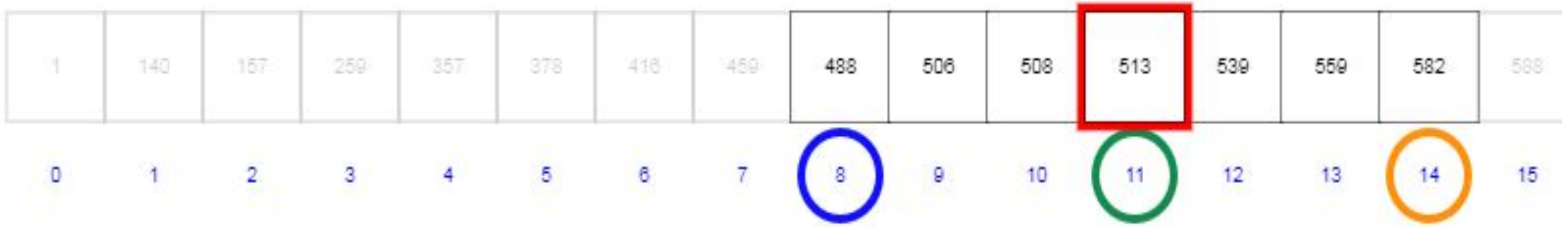
```
def binarySearch(listData, value)
  low = 0
  high = len(listData) - 1
  while (low <= high)
    mid = (low + high) / 2
    if (listData[mid] == value):
      return mid
    elif (listData[mid] < value)
      low = mid + 1
    else:
      high = mid - 1
  return -1
```



```
def binarySearch(listData, value)
  low = 0
  high = len(listData) - 1
  while (low <= high)
    mid = (low + high) / 2
    if (listData[mid] == value):
      return mid
    elif (listData[mid] < value)
      low = mid + 1
    else:
      high = mid - 1
  return -1
```

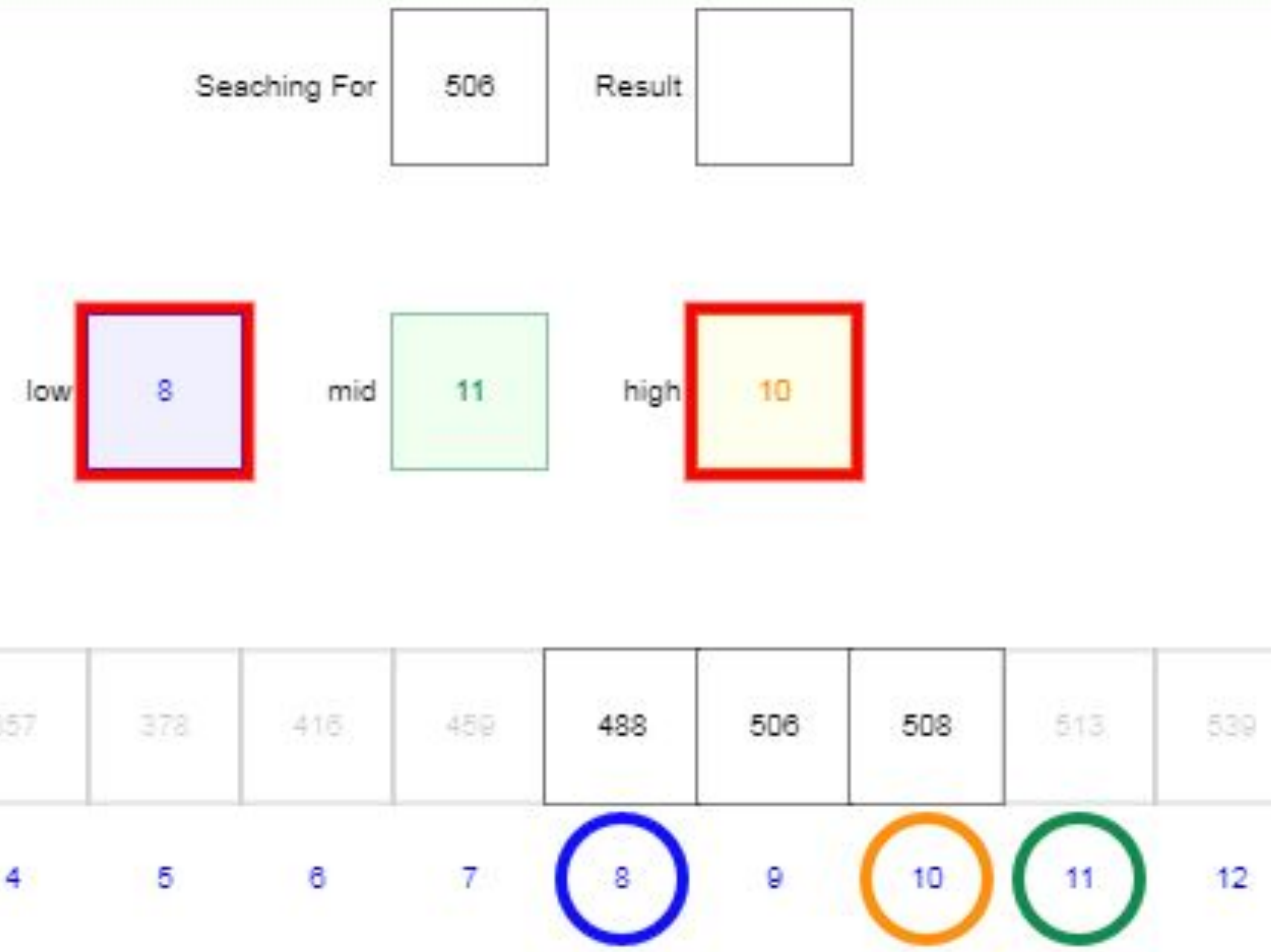
Seaching For $\boxed{506}$ Result $\boxed{}$

low $\boxed{8}$ mid $\boxed{11}$ high $\boxed{14}$

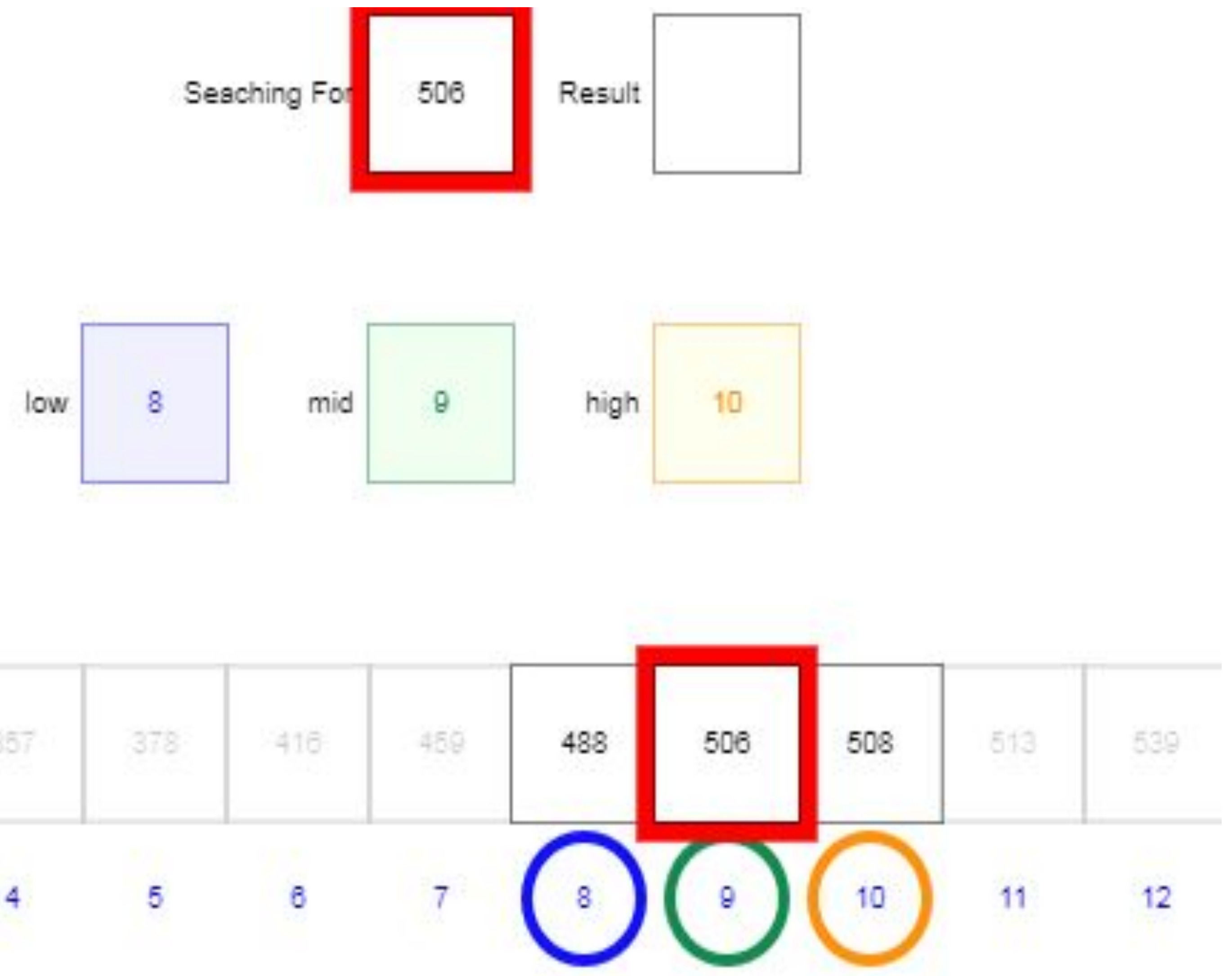


1	140	157	259	357	378	416	459	488	506	508	513	539	559	582	588
0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

```
def binarySearch(listData, value)
    low = 0
    high = len(listData) - 1
    while (low <= high)
        mid = (low + high) / 2
        if (listData[mid] == value):
            return mid
        elif (listData[mid] < value)
            low = mid + 1
        else:
            high = mid - 1
    return -1
```



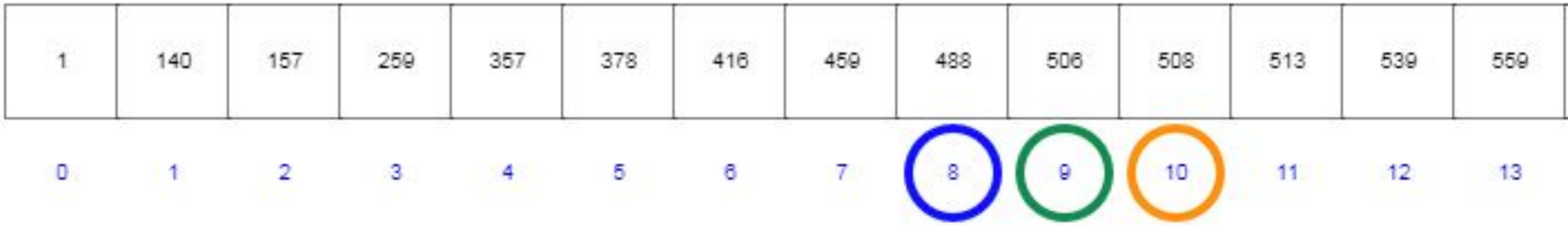
```
def binarySearch(listData, value)
    low = 0
    high = len(listData) - 1
    while (low <= high)
        mid = (low + high) / 2
        if (listData[mid] == value):
            return mid
        elif (listData[mid] < value)
            low = mid + 1
        else:
            high = mid - 1
    return -1
```



```
def binarySearch(listData, value)
  low = 0
  high = len(listData) - 1
  while (low <= high)
     mid = (low + high) / 2
     if (listData[mid] == value):
        return mid
     elif (listData[mid] < value)
        low = mid + 1
     else:
        high = mid - 1
  return -1
```

Seaching For $\boxed{506}$ Result $\boxed{9}$ Element found

low $\boxed{8}$ mid $\boxed{9}$ high $\boxed{10}$



1	140	157	259	357	378	416	459	488	506	508	513	539	559
0	1	2	3	4	5	6	7	8	9	10	11	12	13