Lecture 9-2 : Array Examples
: Bubble Sort and Binary Search

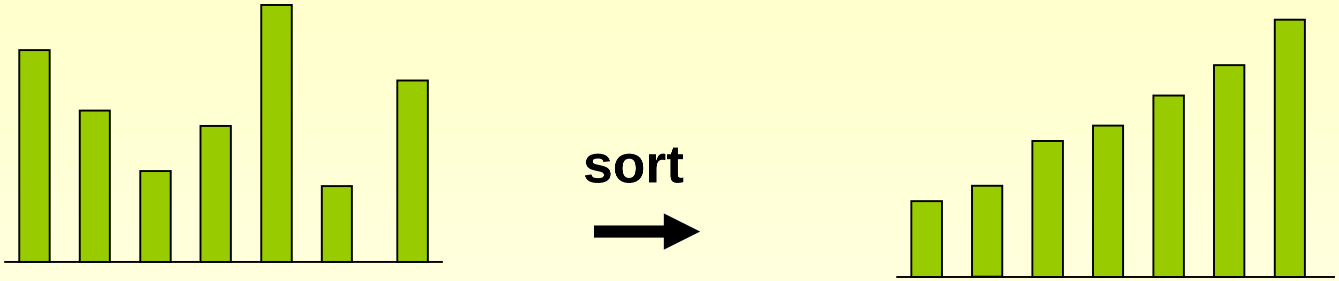
Acknowledgment : Lecture notes from Ohio Supercomputing Center

Array arguments in function

```
#include <stdio.h>

void myFunction(int yourArray[], int n);

int main ()
{
    int i;
    int myArray[4] = {0};
    myFunction( myArray, 4);
    for (i = 0; i < 4; i++)
        printf(“%d\n”,myArray[i]);
    return 0;
}

void myFunction (int yourArray[], int n)
{
    int i;
    for (i = 0; i < n; i++)
        yourArray[i] = i;
}
```

```
0
1
2
3
```

Examples : Sorting , Binary Search



sort

Linear search vs binary search

Search **12**

?

In worst case, 15 comparisons are necessary using lineary search

25 34 7 36 29 91 83 42 6 13 73 54 22 63 12

sort

1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
6	**7**	**12**	**13**	**22**	**25**	**29**	**34**	**36**	**42**	**54**	**63**	**73**	**83**	**91**

If sorted, at most 4 comparisons are enough using binary search.

bubble sorting

- Simple sorting algorithm
- Relatively slow
- Time Complexity
  - Worst case O(N^2)
  - Average case O(N^2)

Step-by-step example sorting (5, 1, 4, 2, 8)

- First Pass :

  (**5 1** 4 2 8) -> (**1 5** 4 2 8)

  (1 **5 4** 2 8) -> (1 **4 5** 2 8)

  (1 4 **5 2** 8) -> (1 4 **2 5** 8)

  (1 4 2 **5 8**) -> (1 4 2 **5 8**)

- Second Pass :

  (**1 4** 2 5 8) -> (**1 4** 2 5 8)

  (1 **4 2** 5 8) -> (1 **2 4** 5 8)
  (1 2 **4 5** 8) -> (1 2 **4 5** 8)

- Third Pass :

  (**1 2** 4 5 8) -> (**1 2** 4 5 8)
  (1 **2 4** 5 8) -> (1 **2 4** 5 8)

- Forth Pass :

  (**1 2** 4 5 8) -> (**1 2** 4 5 8)

Bubble sort function

```
void bubbleSort(int A[], int N)
{
   int temp,i, j;
   for (i=N-1;i>=1;i--)
        for (j=1;j<=i;j++)
               if (A[j-1]>A[j]) {
                       temp=A[j-1];
                       A[j-1]=A[j];
                       A[j]=temp;
               }
}
```

```
#include <stdio.h>

void bubbleSort(int A[], int N);

int main()
{
   int i;
   int a[5]={5,1,4,2,8};
   bubbleSort(a,5);
   for (i=0;i<5;i++)
        printf(“%d\n”,a[i]);

   return 0;
}
```

Binary Search

```
/* This function searches for v in array a
   with size N.
   If found, it returns the array index.
   If not found, it returns -1. */
int binsearch(int a[], int N, int v)
{
   int l=0; int r=N-1; int x;
   while (r>=l) {
        x=(l+r)/2;
        if (v<a[x]) r=x-1;
        else l=x+1;
        if (v==a[x]) return x;
   }
   return -1;
}
```

```
#include <stdio.h>

int binsearch(int a[], int N, int v);

int main()
{
   int position;
   int a[5]={1,2,4,5,8};
   position = binsearch(a, 5, 2);
   printf(“%d\n”,position);

   return 0;
}
```

binary search (recursive function)

```
/* This function searches for v in array a with size N.
   If found, it returns the array index.
   If not found, it returns -1. */
int binsearch(int a[], int N, int v)
{
// insert your code here











}
```

Bubble Sort example 2

1	2	3	4	5	6	7	8
23	**17**	9	12	15	6	34	12

>?

1	2	3	4	5	6	7	8
17	**23**	9	12	15	6	34	12

1	2	3	4	5	6	7	8
17	**23**	**9**	12	15	6	34	12

>?

1	2	3	4	5	6	7	8
17	**9**	**23**	12	15	6	34	12

1	2	3	4	5	6	7	8
17	**9**	**23**	**12**	15	6	34	12

>?

1	2	3	4	5	6	7	8
17	**9**	**12**	**23**	15	6	34	12

1	2	3	4	5	6	7	8
17	**9**	**12**	**23**	**15**	6	34	12

>?

1	2	3	4	5	6	7	8
17	**9**	**12**	**15**	**23**	6	34	12

1	2	3	4	5	6	7	8
17	**9**	**12**	**15**	**23**	**6**	34	12

>?

1	2	3	4	5	6	7	8
17	**9**	**12**	**15**	**6**	**23**	34	12

1	2	3	4	5	6	7	8
17	9	12	15	6	23	34	12

>?

1	2	3	4	5	6	7	8
17	9	12	15	6	23	34	12

≤

1	2	3	4	5	6	7	8
17	9	12	15	6	23	34	12

>?

1	2	3	4	5	6	7	8
17	9	12	15	6	23	12	34

1	2	3	4	5	6	7	8
17	9	12	15	6	23	12	34

>?

1	2	3	4	5	6	7	8
9	17	12	15	6	23	12	34

1	2	3	4	5	6	7	8
9	17	12	15	6	23	12	34

>?

1	2	3	4	5	6	7	8
9	12	17	15	6	23	12	34

1	2	3	4	5	6	7	8
9	12	17	15	6	23	12	34

>?

1	2	3	4	5	6	7	8
9	12	15	17	6	23	12	34

1	2	3	4	5	6	7	8
9	12	15	17	6	23	12	34

>?

1	2	3	4	5	6	7	8
9	12	15	6	17	23	12	34

1	2	3	4	5	6	7	8
9	12	15	6	17	23	12	34

>?

1	2	3	4	5	6	7	8
9	12	15	6	17	23	12	34

1	2	3	4	5	6	7	8
9	12	15	6	17	23	12	34

>?

1	2	3	4	5	6	7	8
9	12	15	6	17	12	23	34

1	2	3	4	5	6	7	8
9	12	15	6	17	12	23	34

>?

1	2	3	4	5	6	7	8
9	12	15	6	17	12	23	34

≤

1	2	3	4	5	6	7	8
9	12	15	6	17	12	23	34

>?

1	2	3	4	5	6	7	8
9	12	15	6	17	12	23	34

≤

1	2	3	4	5	6	7	8
9	12	15	6	17	12	23	34

>?

1	2	3	4	5	6	7	8
9	12	6	15	17	12	23	34

1	2	3	4	5	6	7	8
9	12	6	15	17	12	23	34

>?

1	2	3	4	5	6	7	8
9	12	6	15	17	12	23	34

≤

1	2	3	4	5	6	7	8
9	12	6	15	17	12	23	34

>?

1	2	3	4	5	6	7	8
9	12	6	15	12	17	23	34

1	2	3	4	5	6	7	8
9	12	6	15	12	17	23	34

>?

1	2	3	4	5	6	7	8
9	12	6	15	12	17	23	34

≤

1	2	3	4	5	6	7	8
9	12	6	15	12	17	23	34

>?

1	2	3	4	5	6	7	8
9	6	12	15	12	17	23	34

1	2	3	4	5	6	7	8
9	6	12	15	12	17	23	34

>?

1	2	3	4	5	6	7	8
9	6	12	15	12	17	23	34

>?

1	2	3	4	5	6	7	8
9	6	12	12	15	17	23	34

>?

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

>?

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

>?

1	2	3	4	5	6	7	8
9	6	12	15	12	17	23	34

≤

1	2	3	4	5	6	7	8
9	6	12	12	15	17	23	34

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

≤

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

≤

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

>?

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

≤

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

>?

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

≤

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

>?

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

≤

1	2	3	4	5	6	7	8
6	9	12	12	15	17	23	34

Sorted!