Array Initialization

Example

```
#include <stdbool.h>
#include <stdio.h>
#define N 10
int main() {
    int digit_seen[N] = {false};
    int digit;
    long n;

    printf("Enter a number: ");
    scanf("%ld", &n);

    /* Rest of the programs appears in next slide */
}
```

Array Initialization

Example (contd)

```
while (n > 0) {
    digit = n % 10;
    if (digit_seen[digit])
        break;
    digit_seen[digit] = true;
    n = n/10;
}
if ( n > 0)
     printf("Repeated digit \n");
else
     printf("No repeated digit \n");
```

Array Initialization

Sizeof Operator

- `sizeof(a)` gives size of array in number of bytes.
- Applying `sizeof` on array `int a[10]` gives 40.
- Applying `sizeof` on any sigle element gives 4.
- The number of element can be obtained by `sizeof(a)/sizeof(a[0]) = 10`
- So, loop need not be modied if array length is changed.

```
int a[10];
int i;
for (i = 0; i < sizeof(a)/sizeof(a[0]); i++)
    a[i] = 0;
```

Sorting

Sorting

- Many algorithms exist: **selection sort**, **bubble sort**, **insertion sort**, **counting sort**, **radix sort**, **merge sort**, **quick sort**, etc.
- Two of the common steps involved: input, output.
- Let us design simple functions for the above two steps.

Sorting

Input

- Generate the input by random number generator.
- It needs a seed: we provide current time as seed.

```
void generate(int a[], int n) {
    int i = 0;

    srand((unsigned int) time(NULL)); // Will require time.h
    while (i < n) {
        a[i] = rand() % 100;
        i++;
    }
}
```

Sorting

Output

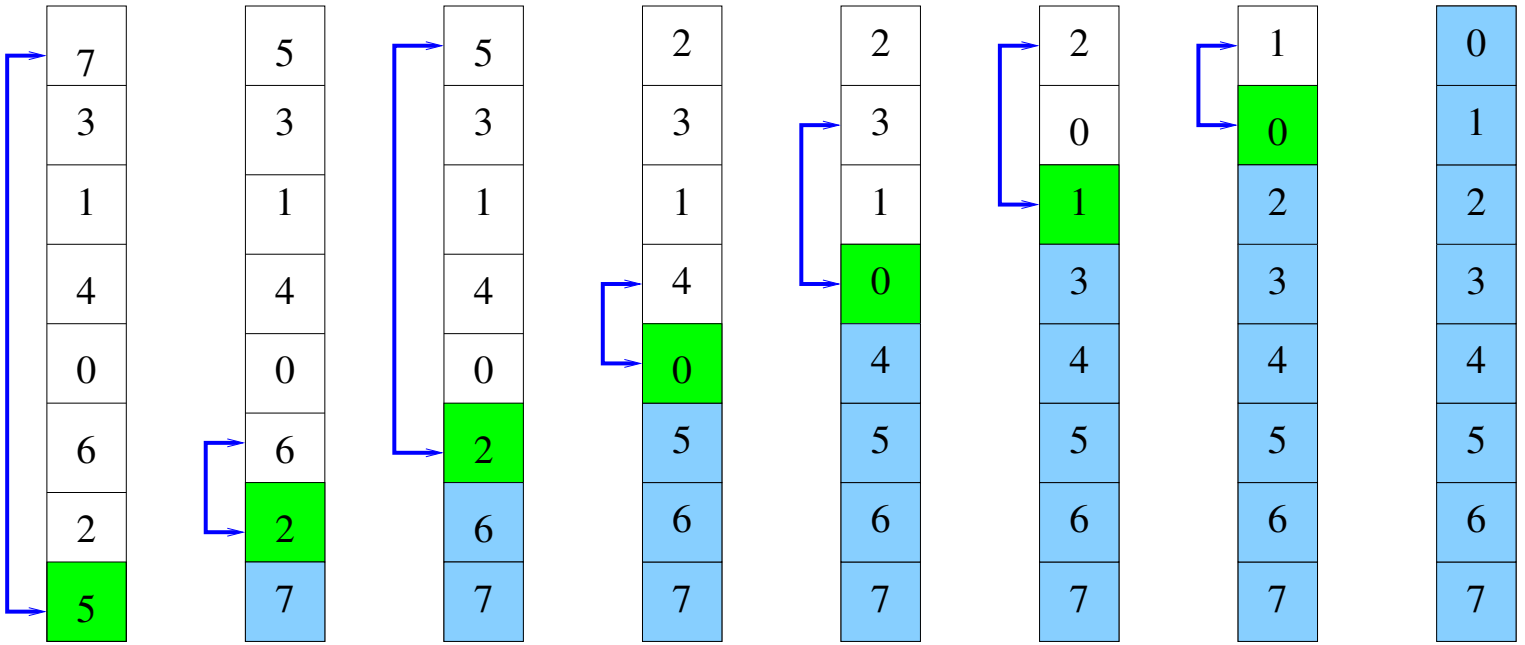
- For output: only a print function is needed.
- It can be used for printing both input/output.

```
void printArray(int a[], int n) {
    int i = 0;
    for (; i < n; i++)
        printf("%d␣␣␣␣", a[i]);
    printf("\n");
}
```

Sorting

Selection Sort

- Find the **largest** element, place it into the **last** position.
- Find the **next largest** element, place it into the **2nd last** position.



Sorting

Selection Sort

```
void selectionSort(int a[], int n) {
    int i, j, max, index;

    for (i = n-1; i >= 0; i--) {
        max = a[i];
        index = i;
        for (j = i-1 ; j >= 0; j--)
            if (a[j] > max) {
                max = a[j];
                index = j;
            }
        a[index] = a[i];
        a[i] = max;
    }
}
```

Sorting

Putting All Together

```
#include <time.h>
#include <stdio.h>
#define N 10
int main() {
    int a[N];
    int n = sizeof(a)/sizeof(a[0]);
    generate(a, n);
    printf("Unsorted input \n");
    printArray(a, n);
    selectionSort(a, n);
    printf("Sorted output \n");
    printArray(a, n);
}
```

Sorting

Bubble Sort

- Bubble sort **sinks** heaviest element to bottom causing the lighter elements to **bubble** up.
- Bubble step is as follow:
  1. Start from the last element in the array,
  2. Compare adjacent pair of array elements, swap if required to push the lighter of pair 1 position up.
  3. Repeat the comparison of next pair until 1st and 2nd element have been compared.
- Sorting is accomplished by repeating **bubble** operation n times.