Array declaration

```
#include <iostream>
using namespace std;
int main() {

  int arr[100]; // declaration of an array
                // of integers of size 100
  // The elements of 'arr' are denoted by
  // arr[0], arr[1], arr[2], ... arr[99]

  // We can assign values to the elements of the array:
  arr[0] = 15;   arr[1] = 26;
  arr[10] = 71;  arr[99] = 27;

  // and we can access them:
  int x = arr[1];
  int y = arr[2*4 + 2];
  arr[98] = arr[99] - arr[0] + 1;
}
```

Array Initialization

```
#include <iostream>
using namespace std;

const int N = 1000;
int main() {

  char c1[5] = {'A', 'T', 'G', 'C', 'T'};

  char c2[5] = {'Z'}; // the first element is set equal
                      // to 'Z', and the other four to '\0'

  char c3[]  = {'@', '#', '$', '%', '^', '&', '*'};
  // here, the size of the declared array is 7, and
  // it's determined from the given list of characters

  char c4[N] = {'A'}; // using an integer constant
                      // to specify the size
}
```

Array Initialization

How to create an array of 25000 integers and initialize it with random values in the interval $0 \leq x \leq 999$?

Array Initialization

How to create an array of 25000 integers and initialize it with random values in the interval $0 \leq x \leq 999$?

```
#include <iostream>
#include <cstdlib>
using namespace std;

const int N = 25000;
int main() {
  srand(time(NULL));

  int arr[N] = {0};
  for(int i = 0; i < N; i++){
    arr[i] = rand() % 1000;
  }
}
```

Searching in an array

Design a program that given an array of integers, decides whether or not the array contains the number 123.

Searching in an array

Design a program that given an array of integers, decides whether or not the array contains the number 123.

```
bool contains = false;

for(int i = 0; i < N; i++) {
  if (arr[i] == 123) {
    contains = true;
    break;
  }
}

// contains == true or false
```

We can define a function for doing this task →

Searching in an array

Design a program that given an array of integers, decides whether or not the array contains the number 123.

A solution in the form of a function:

```
const int N = 25000;

bool contains_123(int arr[N]) {

  for(int i = 0; i < N; i++) {
    // if 123 is found, return true immediately
    if (arr[i] == 123) return true;
  }
  // otherwise, if 123 was not found:
  return false;
}
```

Searching for the largest element

Design a program that finds the largest element of a given array of integers.

Searching for the largest element

Design a program that finds the largest element of a given array of integers.

```
const int N = 25000;

// Finds the largest element of the array
// It assumes that the array length N >= 1.
int largest(int arr[N]) {

  int max = arr[0];
  for(int i = 0; i < N; i++) {
    if (arr[i] > max) { max = arr[i]; }
  }
  return max;

}
```

Reverse an array

Design a program that reverses an array of integer.

Reverse an array

Design a program that reverses an array of integer.

```
const int N = 5000;

// swaps elements i and j
void swap(int arr[N], int i, int j) {
  int tmp = arr[i];
  arr[i] = arr[j];
  arr[j] = tmp;
}
// reverse the array (updates the array in-place)
void reverse(int arr[N]) {
  for(i = 0; i < N/2; i++) {
    swap(arr, i, N-i);
  }
}
```

Good and bad about arrays

Good:

- Storage for indexed data, where all elements have the same type.
- Can create multidimesional arrays (arrays of arrays of arrays . . .).
- Can pass an array as an argument to a function (we will talk more about it later).

Bad:

- Cannot resize an array (..yet. First, need to learn dynamic memory allocation). For now, we will be declaring our arrays just big enough to accomodate the input.
- Cannot return an array from a function (As any other variable, an array gets "destroyed" when it goes out of the scope) Again, dynamic memory allocation will resolve this issue.

Compacting an array of integers

Given an array of integers that may contain several zeroes, for example

$$\{0, 5, 2, 0, 0, 7, 1, 0, 1\}$$

Design a program that pushes all non-zero elements to the beginning of the array, preserving their original order. The array gets updated in place:

$$\{5, 2, 7, 1, 1, 0, 0, 0, 0\}$$

(Note that we may consider a similar algorithm, where we actually create a new compacted array, instead of changing the original one. This aproach would preserve the original data, which may be useful. On the other hand, it would use extra memory, which is not necessarily what we want)