# Searching for the largest element

```
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;

}
```

# Searching for the largest element

We can define the same function for arrays of arbitrary size. Because the size of the array is not provided in the type of the array parameter `arr[]`, we pass another parameter `len` with the size of the array.

```
// Find the largest element, defined for
// arrays of any size
int largest(int arr[], int len) {

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

}
```

# Print the elements of the array

```
// Print the array of integers
void print(int arr[], int len) {

  for(int i = 0; i < len; i++) {
    cout << arr[i] << ' ';
  }
  cout << endl;

}

int main() {
  int data1[10] = {1, 2, 4, 8, 16, 32, 64, 128, 256, 512};
  print (data1, 10);

  int data2[4] = {48, 37, 59, 40};
  print (data2, 4);
}
```

# Does an array of integers contain $X$?

We want to write a function

```
bool contains(int arr[], int len, int x)
```

that returns `true` if `x` is in the array, otherwise return `false`.

# Does an array of integers contain $X$?

We want to write a function

```
bool contains(int arr[], int len, int x)
```

that returns `true` if `x` is in the array, otherwise return `false`.

```
// returns true if arr contains x
bool contains(int arr[], int len, int x) {

  for(int i = 0; i < len; i++) {
    if (arr[i] == x) return true;
  }
  return false;

}
```

# Unique integers

Ask the user to input 10 integers, and store in an array those that are unique. Then print them out.

# Unique integers

```
int main() {

  int arr[N] = {0}; // the array to store numbers
  int len = 0; // the length of the array

  for(int n = 0; n < N; n++) {
    int input = 0;
    cout << "Please enter a number: ";
    cin >> input;
    if (!contains(arr, len, input)) {
      arr[len] = input;
      len++;
    }
  }

  print(arr, len);

}
```

# Replacing all occurrences of $x$ with $y$

We want to write a function that replaces all occurrences of $x$ with $y$ in the given array.

# Replacing all occurrences of $x$ with $y$

We want to write a function that replaces all occurrences of $x$ with $y$ in the given array.

```
// replaces all occurrences of x with y
void replace(int arr[], int len, int x, int y);
```

# Replacing all occurrences of $x$ with $y$

We want to write a function that replaces all occurrences of $x$ with $y$ in the given array.

```
// replaces all occurrences of x with y
void replace(int arr[], int len, int x, int y) {

  for(int i = 0; i < len; i++) {
    if (arr[i] == x)
      arr[i] = y;
  }

}
```

# Searching for $X$ in the array

The function should search for the value $X$ in the array, and if it's found, return its index.

For example, given

```
int data[5] = {12, 65, 21, 25, 11},
```

We ask the function: Where is 25?

The answer should be: 3

# find: Implementation I

```
#include <iostream>
const int NOT_FOUND = -1;

// Returns the index if the first occurrence of x in arr.
// If there is none, returns -1 (which means not found)
int find(int arr[], int len, int x) {
  for(int i = 0; i < len; i++) {
    if (arr[i] == x) return i;
  }
  return NOT_FOUND;
}

int main() {
  int data[5] = {1, 7, 3, 95, 11};
  int i = find(data, 5, 95);
  if (i != NOT_FOUND)
    std::cout << "Found at " << i << std::endl;
}
```

# find: Implementation II

```
#include <iostream>
// If arr[i] == x for some i, return true and set index = i,
// otherwise return false.
bool find2(int arr[], int len, int x, int &index) {
  for(int i = 0; i < len; i++) {
    if (arr[i] == x) {
      index = i;
      return true;
    }
  }
  return false;
}
int main() {
  int data[5] = {1, 7, 3, 95, 11};
  int i = 0;
  if (find2(data, 5, 95, i))
    std::cout << "Found at " << i << std::endl;
}
```

# How to sort an array?

# How to sort an array?

```
void sort(int arr[], int size) {
// we iteratively make our array sorted, first we put
// the smallest element at the front, then find the second
// smallest element, put it in the second place,
// and keep doing so until there is no elements left.

  for (int i = 0; i < size; i++) {
    // find the smallest element
    int index_min = i;
    for (int j = i+1; j < size; j++) {
      if (arr[index_min] > arr[j]) {
        index_min = j;
      }
    }
    // and put it in front
    swap(arr[i], arr[index_min]);
  }
}
```

# How to shuffle an array?

# How to shuffle an array?

Read about Fisher-Yates's algorithm.