Topic 3

1. Arrays
2. Common array algorithms
3. Arrays / functions
4. Problem solving: adapting algorithms
5. Problem solving: discovering algorithms
6. 2D arrays
7. Vectors
8. Chapter Summary

Arrays as Parameters in Functions

Recall that when we work with arrays
we use a companion variable.

The same concept applies when
using arrays as parameters:

You must pass the size to the function
so it will know how many elements to work with.

Array Parameter Function Example

Here is the **sum** function with an array parameter:
Notice that to pass one array, it takes two parameters.

```
   double sum(double data[], int size)
{
    double total = 0;
    for (int i = 0; i < size; i++)
    {
       total = total + data[i];
    }
    return total;
}
```

Array Parameters in Functions Require `[]` in the Header

You use an empty pair of square brackets *after* the parameter variable's name to indicate you are passing an array.

```
double sum(double data[], int size)
```

Array Function Call Does NOT Use the Brackets!

When you call the function, supply both the name of the array and the size, BUT NO SQUARE BRACKETS!!

```
double NUMBER_OF_SCORES = 10;
double scores[NUMBER_OF_SCORES]
   = { 32, 54, 67.5, 29, 34.5, 80, 115, 44.5, 100, 65 };
double total_score = sum(scores, NUMBER_OF_SCORES);
```

You can also pass a smaller size to the function:

```
double partial_score = sum(scores, 5);
```

This will sum over only the first five **`doubles`** in the array.

Array Parameters Always are Reference Parameters

When you pass an array into a function, the contents of the array can ***always*** be changed. An array name is actually a reference, that is, a memory address:

```
   //function to scale all elements in array by a factor
void multiply(double values[], int size, double factor)
  {
     for (int i = 0; i < size; i++)
     {
        values[i] = values[i] * factor;
     }
  }
```

But never use an & with an array parameter – that is an error.

Arrays as Parameters but No Array Returns

You can pass an array into a function
but

you cannot return an array.

However, the function can modify an input array, so the function definition must include the result array in the parentheses if one is desired.

Arrays as Parameters and Return Value

If a function can change the size of an array, it should let the caller know the new size by returning it:

```
int read_inputs(double inputs[], int capacity)
{ //returns the # of elements read, as int
   int current_size = 0;
   double input;
   while (cin >> input)
   {
      if (current_size < capacity)
      {
         inputs[current_size] = input;
         current_size++;
      }
   }
   return current_size;
}
```

Array Parameters in Functions: Calling the Function

Here is a call to the **`read_inputs`** function:

```
const int MAXIMUM_NUMBER = 1000;
double values[MAXIMUM_NUMBER];
int current_size =
  read_inputs(values, MAXIMUM_NUMBER);
```

After the call, the **`current_size`** variable specifies how many were added.

Function to Fill or Append to an Array

Or it can let the caller know by using a reference parameter:

```
void append_inputs(double inputs[], int capacity,
                   int& current_size)
{
   double input;
   while (cin >> input)
   {
      if (current_size < capacity)
      {
         inputs[current_size] = input;
         current_size++;
      }
   }
}
```

Note this function has the added benefit of either filling an empty array or appending to a partially-filled array.

Array Functions Example

The following program uses the preceding functions to read values from standard input, double them, and print the result.

- The **`read_inputs`** function fills an array with the input values. It returns the number of elements that were read.
- The **`multiply`** function modifies the contents of the array that it receives, demonstrating that arrays can be changed inside the function to which they are passed.
- The **`print`** function does not modify the contents of the array that it receives.

Array Functions Example Code, Part 1

```
#include <iostream>
using namespace std;
//ch06/functions.cpp
/**
Reads a sequence of floating-point numbers.
@param inputs an array containing the numbers
@param capacity the capacity of that array
@return the number of inputs stored in the array
*/
int read_inputs(double inputs[], int capacity)
{
   int current_size = 0;
   cout << "Please enter values, Q to quit:" << endl;
   bool more = true;
   while (more)
   {
```

Array Functions Example Code, part 2

```
      double input;
       cin >> input;
       if (cin.fail())
       {
          more = false;
       }
       else if (current_size < capacity)
       {
          inputs[current_size] = input;
          current_size++;
       }
   }
   return current_size;
}
```

Array Functions Example Code, part 3

```
/** Multiplies all elements of an array by a factor.
@param values a partially filled array
@param size the number of elements in values
@param factor the value with which each element is multiplied */
void multiply(double values[], int size,
              double factor)
{
   for (int i = 0; i < size; i++)
   {
      values[i] = values[i] * factor;
   }
}
/** Prints the elements of a vector, separated by commas.
@param values a partially filled array
@param size the number of elements in values */
void print(double values[], int size)
{
   for (int i = 0; i < size; i++)
   {
      if (i > 0) { cout << ", "; }
      cout << values[i];
   }
   cout << endl;
}
```

Array Functions Example Code, part 4

```
int main()
{
   const int CAPACITY = 1000;
   double values[CAPACITY];
   int size = read_inputs(values, CAPACITY);
   multiply(values, size, 2);
   print(values, size);

   return 0;
}
```

Constant Array Parameters

- When a function doesn’t modify an array parameter, it is considered good style to add the `const` reserved word, like this:

```
double sum(const double values[], int size)
```

- The `const` reserved word helps the reader of the code, making it clear that the function keeps the array elements unchanged.
- If the implementation of the function tries to modify the array, the compiler issues a warning.