Arrays / Vectori (in romanian)

- You will often need to store many data values of a particular kind in your programs
  - For example we might want to keep track of the heights of our colleagues and then compute an average
- we can easily solve this kind of problem by creating as many variables as needed. For example, if we only have 5 colleagues we can create 5 variables as follows:

```
int heightAndrei = 175;
int heightBogdan = 172;
int heightRoxana = 165;
int heightMirela = 178;
int heightIoana = 199;
```

- But as you might observer, this can become troublesome if we have more data that we want to store.
  - What will happen if we want to save the height of our 40 colleagues? Should we define 40 variables? No way!
- In this lesson we will see how can we solve this problem using Arrays (Vectors)

Introduction

- An array is a fixed number of data items that are all of the same type.
- The data items in an array are referred to as elements
- The folowing array declaration is similar to a declaration for a normal variable that contains a single value, except that you've placed a number between square brackets `[]` following the name:
  - ```
    long numbers[10];
    ```
  - The number between square brackets defines how many elements the array contains and is called the array dimension
  - Each of the data items stored in an array is accessed by the same name
    - In the previous statement the array name is `numbers`
  - You select a particular element by using an `index value` between square brackets following the array name
  - Index values are sequential integers that start from zero, and `0` is the index value for the first array element
    - In our example, the index values for the elements in the `numbers` array run from 0 to 9, so the index value 0 refers to the first element and the index value 9 refers to the last element
    - Therefore you access the elements in the `numbers` array as `numbers[0], numbers[1], numbers[2], etc...`
  - Always remember that index values start from `0` and not `1`. In our case, the third element can be accessed using `numbers[2]`, due to this rule of starting from `0`.
    ```

    - You can think of the index value for an array element as the offset from the first element:
        - The first element, is the first elements thus the offset is 0
        - The second element is offset by 1 from the first element, thus we access it via `numbers[2]`
        - etc...

- We can also specify an index for an array element by an expression in the square brackets following the array name. The expression must result in an integer value that correspond to one of the possible index values.

    - For example you could write numbers `[i-2]`. If i is 3m this accesses `numbers[1]`, the second element in an array.
    - The only constraint for using an expression when accessing an element is that it must produce an `integer` result, and the result must be a legal index value for the array
        - If you use an expression for an inex value that's outside the legal range for the array, the program will give undefined results. The compiler cannot help in this situation and the program will pick whatever garbage resides at that memory For a better understanding of arrays, we should start with a example. In the following snippet we will see how easy is to compute the average grade score for the students in a class.

- Class Exercise 1

    - Compute the average grade of students from a class with 7 students. Each grade should be read via the standard input. The average should be computed using the basic arithmetic average formula.
    - Solution:

        - ```
          #include <iostream>

          using namespace std;
          int main() {

              int grade = 0;
              int numberOfStudents = 7;
              int sum = 0;
              float average = 0.0f;

              for (int i = 0; i < numberOfStudents; i++) {
                  cout << "Enter a grade:";
                  cin >> grade;
                  sum += grade;
              }
              average =  (float)sum / numberOfStudents;
              cout << "\nAverage of the grades from these 7 students is: "
          << average;

              return 0;
          }
          ```
          ```

    - Note: as we were only interested in finding the average, we had no reason to also save the grades somewhere

- Class Exercise 2:

    - Building on the previous exercises, consider that we also want to save the grades for each student and then display the grade and the average next to it. In this scenarion, we will make use of arrays
    - Solution:

```
#include <iostream>

using namespace std;
int main() {
    int grades[10];
    int count = 10;
    int sum = 0;
    float average = 0.0f;
    cout << "Enter the 10 grades: \n";
    for (int i = 0; i < count; i++) {
        cout << i+1  <<"> ";
        cin >> grades[i];
        sum += grades[i];
    }
    average = (float) sum / count;
    for(int i = 0; i < count; i++) {
        cout<< "Student #" << i+1 <<": " << grades[i] <<" average
is : " << average << endl;
    }
    return 0;
}
```

Arrays and memory management

- When you declare an array, you give the compiler all the information it needs to allocate the memory for it.
- The type of value determines the number of bytes that each element will require
- The array dimension specifies the number of elements
- The number of bytes that an array will occupy is the number of elements multiplied by the size of each element
- One important trick from the formula above, in order to find how many elements are inside an array, we simply divide the size of the entire array by the size of the first element:

    - ```
      cout << sizeof(grades) / sizeof(grades[0]);
      ```

    - The explanation here is as follows:
      ```

    - As we have 10 elements in our array, `sizeof(grades)` is `10 * sizeof(int)` which on my machine translates to `10 * 4` as int occupies 4 bytes on my machine
    - And `sizeof(grades[0])` is 4 because `grades[0]` is an `int` number and `sizeof(int)` is `4 bytes`
    - Now dividing `40 / 4` gives us the total number of elements in the array

Initializing an array

- Most of the time we will want to assign initial values for the elements of your array most of the time, even if it's only for safety's sake.
- To initialize the elements of an array, you just specify the list of initial values between braces and separate them by commas in the declaration:

```
double values[5] = {1.5, 2.5, 3.4, 4.3, 5.1};
```

- To initialize the whole array, there must be one value for each element. If there are fewer initializing values than elements, the elements without initializing values will be set to 0. This means that if you write

```
double values[5] = {1.5, 2.5, 3.5};
```

  only the first three elements will have values and the rest will be set to zero; The final array will look like this:

```
Index   0       1       2       3       4       5
Value   1.5     2.5     3.5     0       0       0
```

- if you put more initializing balues than there are array elements, you'll get an error message from the compiler
- However, you can omit the size of the array when you specify a list of initial values. In this case, the compiler will assume that the number of elements is the number of values in the array:

```
int primes[] = { 2, 3, 5, 7, 11, 13, 17, 19, 23, 29};
```

    - The size of the array is determined by the number of initial values in the list, so the primes array will have ten elements

Major pitfalls

- We can always make the mistake of iterating past the limits of an array, meaning that, for example we can try to iterate until 12 even though our array has only 10 elements. See the snippet below:

```
#include <iostream>

using namespace std;
int main() {
    int grades[10];
    for(int i = 0; i < 10; i++) {
        grades[i] = i * 10;
    }

    for(int i =0; i < 12; i++) {
        cout << grades[i] << endl;
    }
}
```

  - THe first loop initializes the array, by multiplying each element with 10 and then, we iterate again over the values but we set 12 as the limit even though we only have 10 values
  - Most of the time, the program will work but will display whatever garbage finds at that address

- Class Exercises 3:
  - Read 13 digits (0 to 9 inclusive) from stdin and then display their histogram
  - Solution:

```
#include <iostream>
using namespace std;
int main() {
    int histogram[10] = {0};
    for(int i = 0; i < 13; i++) {
        cout << "Enter a number between 0 and 9: ";
        int number;
        cin >> number;
        histogram[number]++;
    }
    for (int i=0; i < 10; i++) {
        cout<< "We have seen " << i << ": " << histogram[i] << "
time(s)" << endl;
    }
}
```

- Class Exercise 4:
  - Write a C++ program which sorts an array in ascending order.
  - Solution:

```
#include <iostream>
```

```
using namespace std;
int main() {
    int numbers[] =  {1,2,5,3,1,5,1,8,0,5,3,5};
    int numbersLength = sizeof(numbers) / sizeof(numbers[0]);
    for(int i = 0; i < numbersLength;i++) {
        for (int j = 0; j < numbersLength-1; j++) {
            if (numbers[j] > numbers[j+1]) {
                int temp = numbers[j];
                numbers[j] = numbers[j+1];
                numbers[j+1] = temp;
            }
        }
    }

    //display the array
    for (int i = 0; i < numbersLength; i++) {
        cout << numbers[i] << " ";
    }
}
```

- Class Exercises 5:
  - Write a C++ program which merges two arrays of the same size;
  - Solution:

```
#include <iostream>

using namespace std;
int main() {
    int numbers[] =  {1,2,3,4,5,6};
    int numbers2[] = {7,8,9,10,11,12};
    int numbersLength = sizeof(numbers) / sizeof(numbers[0]);
    int result [2 *numbersLength];

    // Add numbers from first array
    for(int i = 0; i < numbersLength;i++) {
    result[i] = numbers[i];
    result[i+numbersLength] = numbers2[i];
    }

    //display the array
    for (int i = 0; i < 2*numbersLength; i++) {
        cout << result[i] << " ";
    }
}
```

- Class Exercise 6:
  - Write a C++ program which deletes all the duplicates from an array.
  - Solution:

```
#include <iostream>

using namespace std;
int main() {
    int numbers[] =  {1,2,1,2,1,2,1,2,1,2};
    int numbersLength = sizeof(numbers) / sizeof(numbers[0]);
    for(int i =0; i < numbersLength; i++) {
        for (int j = i+1; j < numbersLength; j++) {
            if(numbers[i] == numbers[j]) {
                for (int k = j; k < numbersLength; k++) {
                    numbers[k] = numbers[k+1];
                }
                numbersLength--;
                j--;
            }
        }
    }
    for (int i = 0; i < numbersLength; i++) {
        cout << numbers[i] << " ";
    }
}
```

- Class Exercise 7:
  - Write A C++ program to find K largest elements in a given array of integers
  - Solution:

```
#include <iostream>

using namespace std;
int main() {
    int numbers[] =  {1,3,2,6,4,8,5,9};
    int numbersLength = sizeof(numbers) / sizeof(int);
    int k;
    cout << "Enter K: ";
    cin >> k;

    for (int i = 0; i < numbersLength; i++) {
        for (int j = 0; j < numbersLength-1; j++) {
            if (numbers[j] < numbers[j + 1]) {
                int temp = numbers[j];
                numbers[j] =  numbers[j+1];
                numbers[j+1] = temp;
            }
        }
    }

    for (int i =0; i < k; i++) {
        cout << numbers[i] << " ";
```

```
        }
    }
```

- Advanced:
  - For extra points, give a K greater then the length of the array
  - Find a solution for the previous bug.

Homework exercises:

1. Write a program in C++ to store 10 elements in an array and then print it.
   - Sample input: Input 10 elements in the array: element #1: 1 element #2: 4 .... element #10: 99
   - Sample output: Elements in array are: 1 4 ... 99
2. Write a program in C++ to read 10 number of values in an array and display it in reverse order
   - Sample input: Input 10 elements in the array: element #1: 1 element #2: 4 element #3: 5 .... element #10: 99
   - Sample output: Elements in reversed order are: 99... 5 4 1
3. Write a program in C++ which finds the sum of all elements of an array with 10 elements:
   - Sample input: Input 10 elements in the array: element #1: 1 element #2: 4 element #3: 5 .... element #10: 99
   - Sample output: Sum is: 66 (note you will have a different value as this sum is for display only)
4. Write a C++ program which will read an array and then display how many numbers have exactly 2 appearance in it.
   - Sample input:
     - 1 12 2 12 1 1 7 3 5 6 7
   - Sample output:
     - 2 (only 7 and 12 appear exactly 2 times in the array)
5. Write a C++ program to print all unique elements in an array
   - Sample input:
     - 1 2 3 4 2 3 5 6 7 8
   - Sample output:
     - Unique elements are: 1 4 5 6 7 8
6. Write a C++ program to merge two arrays of same size and then sort them in descending order.
   - Sample input:
     - 1 2 3 4
     - 5 6 7 8
   - Sample output:
     - 8 7 6 5 4 3 2 1
7. Write a C++ program to count total number of duplicate elements in an array.

- Sample input:
  - 1 2 3 1 2 23 4 3
- Sample output: There are 3 duplicates

8. Write a C++ program to find maximum and minimum element in an array

- Sample input:
  - 3 2 1 -3 5 9 7
- Sample output:
  - Min is -3 and Max is 9

9. Write a C++ program to find second largest element in an array

- Sample input:
  - 3 2 1 -3 5 9 7
- Sample output:
  - Second largest number is 7