Why do we need an array?

Suppose you want to record the scores of all students in a class and count the number of students who scored more than the class average. Clearly we need to compute the average before we could count the number of students that scored more than the average. You can compute the average using the following program:

```c
#define NumOfStudents 10
main()
{
    int score, counter, sum;
    float average;
    sum = 0;
    for(counter=1; counter<=NumOfStudents; counter++)
    {
        cout << "Enter a score: ";
        cin >> score;
        sum = sum + score;
    }
    average = (float) sum/NumOfStudents;
    cout << "Average is: " << average << endl;
    return 0;
}
```

Now to find the number of students who scored more than the average, you need to compare each score with the average and count the ones that are larger; but to do that you have to ask the user to enter the scores one more time. That is not an efficient way to complete the task on hand. It would have been better if we had stored all the values when the user entered them. That means, we have to declare (NumOfStudents)10 integers. We could declare them as follows:

```c
int x1, x2, x3, x4, x5, x6, x7, x8, x9, x10;
```

But what if NumOfStudents is 100? You can see how tedious it would be to declare that many integers. Array provides us with an elegant way to declare that many integers:

```c
int x[NumOfStudents];
```

If NumOfStudents is 10, then the 10 integers are x[0], x[1], x[2], x[3], …, x[8], x[9]. If NumOfStudents is 100, then the 100 integers are x[0], x[1], …, x[98], x[99].

We will complete the problem given at the beginning after we learn and understand the basic operations on an array: Declare, Declare and Initialize, Assign, Read and Print.
Processing an array of integers is very similar to processing an integer. We just have to remember that each integer location in an array has a unique name \( x[i] \), where \( i \) is the index of the array location. We can use \( i \) as the counter for the array index in a for-loop and repeat a statement for all integer locations in the array.

- **Given the declaration:** \( \text{int } x[10]; \)

- **A Pictorial Representation of the Array:**

- **Overview of Five Basic Operations:**

<table>
<thead>
<tr>
<th>Operation</th>
<th>One Integer</th>
<th>Array of Integers</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declare</td>
<td>( \text{int } x; )</td>
<td>( \text{int } x[10]; )</td>
</tr>
<tr>
<td>Declare and Initialize</td>
<td>( \text{int } x = 2; )</td>
<td>( \text{int } x[10] = {1,3,6,7,0,8,9,2,4,12}; )</td>
</tr>
</tbody>
</table>
| Store or Assign      | \( x = 2; \) | \( \text{for}(i=0; i<10; i++) \{
  \quad x[i] = 4;
\} \) |
| Read                 | \( \text{cout} \ll \text{“Enter an integer: ”}; \)
\( \text{cin} \gg x; \) | \( \text{for}(i=0; i<10; i++) \{
  \quad \text{cout} \ll \text{“Enter an integer: ”}; \\
  \quad \text{cin} \gg x[i]; \\
\} \) |
| Print                | \( \text{cout} \ll x; \) | \( \text{for}(i=0; i<10; i++) \{
  \quad \text{cout} \ll x[i]; \\
\} \) |

- **Different ways to Declare and Initialize:**
  - \( \text{int } x[10] = \{1, 4, 8, 0, 6, 2, 9, 7, 3, 5\}; \)
    The 10 values are stored in the 10 array locations.
  - \( \text{int } x[10] = \{1, 4, 8, 0, 6, 2\}; \)
    The 6 values are stored in the first 6 array locations. The remaining 4 locations get the value 0.
  - \( \text{int } x[10] = \{1\}; \)
    The first location gets the value 1, the remaining locations get the value 0.
  - \( \text{int } x[] = \{1, 4, 8, 0, 6, 2, 9\}; // \text{the size of the array is missing} \)
    The size of the array is missing! In such a case, the number of values given in the initialization part, determines the size of the array. Hence in this case the size of the array is 7.
Programs using Array of Integers

1) Write a program that reads and stores 10 integers in an array. Print the values stored in the array.
2) Write a program that chooses randomly 20 integers from the range \(-100\) to 100 and assigns them into an array. Print the array in a 2 x 10 format.

```c
#define NumOfIntegers 10
main()
{
    int x[NumOfIntegers], i;
    for(i=0; i<NumOfIntegers; i++)
    {
        cout << "Enter an integer: ";
        cin >> x[i];
    }
    for(i=0; i<NumOfIntegers; i++)
    {
        cout << x[i] << " ";
    }
    cout << endl;
    return 0;
}
```

```c
#define NumOfIntegers 20
main()
{
    int x[NumOfIntegers], i;
    srand(time(NULL));
    for(i=0; i<NumOfIntegers; i++)
    {
        x[i] = rand()%201-100;
    }
    for(i=0; i<NumOfIntegers; i++)
    {
        if (i%10 == 0)
            cout << endl;
        cout << x[i] << " ";
    }
    cout << endl;
    return 0;
}
```

3) Write a program that assigns into an array the first 15 terms in the sequence: 1, 5, 9, 13, 17, 21, 25, .... Print the sequence in a 3 x 5 format.

```c
#define NumOfIntegers 15
main()
{
    int x[NumOfIntegers], i, j;
    for(i=0, j=1; i<NumOfIntegers; i++, j=j+4)
    {
        x[i] = j;
    }
    for(i=0; i<NumOfIntegers; i++)
    {
        if (i%5 == 0)
            cout << endl;
        cout << setw(5) << x[i];
    }
    cout << endl;
    return 0;
}
```

Scratch work to determine the counters:

We want to store:

\[ x[0] = 1; \]
\[ x[1] = 5; \]
\[ x[2] = 9; \]
\[ x[3] = 13; \]
\[ \ldots \]
\[ x[14] = \ldots \]

So we need a counter \( i \) for the array index 0, 1, 2, 3, ..., 14

and another counter \( j \) for the values on the right side 1, 5, 9, ...

For each counter note its initial value and how it increases.
4) Write a program that reads and stores the scores of 20 students in an array. It computes the average and counts the number of students who scored more than the class average. After printing the array in a 4x5 format, print the average and the number of students who got above the average.

```c
#define NumOfStudents 20
main()
{
    int score[NumOfStudents];
    int i, sum, AboveAvg;
    float average;

    // Read the scores into an array and find the sum.
    sum = 0;
    for (i=0; i<NumOfStudents; i++)
    {
        cout << "Enter a score: ";
        cin >> score[i];
        sum = sum + score[i];
    }
    average = (float) sum/NumOfStudents;

    // Print the array and find the number of students who got above average.
    AboveAvg = 0;
    for (i=0; i<NumOfStudents; i++)
    {
        if (i%5==0)
            cout << endl;
        cout << setw(5) << score[i];
        if (score[i] > average)
            AboveAvg = AboveAvg + 1;
    }
    cout << "Average is: 
" << average << endl;
    cout << "Number of students who got score above the average is 
" << AboveAvg << endl;
    return 0;
}
```

Note: Make sure required library files are included in each program. Include stdlib.h when using rand() function. Include iomanip.h when formatting the placement of integers using setw.
An array of characters can be processed the same way as an array of integers. But by making an array of characters into a “string” we can process it like a simple data type (such as a single integer) using functions defined in the library file *string.h*. Hence it is always efficient to make an array of characters as a string and inserting the null character ‘\0’ as the last character of the string does it.

- **Overview of Five Basic Operations:**

<table>
<thead>
<tr>
<th></th>
<th>Array of Characters</th>
<th>To process as a string</th>
</tr>
</thead>
<tbody>
<tr>
<td>Declare</td>
<td><code>char x[10]</code></td>
<td><code>char x[10]</code></td>
</tr>
<tr>
<td>Declare &amp; Initialize</td>
<td><code>char x[10] = {'a','b','c'};</code></td>
<td><code>char x[10] = {'a', 'b', 'c', '\0'};</code></td>
</tr>
<tr>
<td>Store or Assign</td>
<td><code>for(i=0; i&lt;10; i++) { x[i] = 'a'; }</code></td>
<td><code>for(i=0; i&lt;9; i++) { x[i] = 'a'; } x[9] = '\0'; // Last character is a null character</code></td>
</tr>
<tr>
<td>Read</td>
<td><code>for(i=0; i&lt;10; i++) { cout &lt;&lt; &quot;Enter a letter: &quot;; cin &gt;&gt; x[i]; }</code></td>
<td><code>cout &lt;&lt; &quot;Enter a string: &quot;; cin &gt;&gt; x; // Read it as a string</code></td>
</tr>
<tr>
<td>Print</td>
<td><code>for(i=0; i&lt;10; i++) { cout &lt;&lt; x[i]; }</code></td>
<td><code>// if x is a string then print it as a string cout &lt;&lt; x &lt;&lt; endl;</code></td>
</tr>
</tbody>
</table>

- **Different ways to Declare and Initialize:**
  - `int x[10] = {'s', 'm', 'i', 't', 'h'};`
    The 5 letters are stored in the first 5 array locations. The remaining 5 locations have unspecified (“garbage”) values.
  - `int x[10] = {'s', 'm', 'i', 't', 'h', '\0'};`
    The 6 letters are stored in the first 6 array locations. The remaining 4 locations have unspecified values. But *x* is a string “smith” since the last character stored is a null character ‘\0’.
  - `int x[10] = "smith";`
    Since a string is stored in the array, the first 5 locations have the letters in “smith” and the 6th location has ‘\0’.
  - `int x[] = "smith"; // the size of the array is missing`  
    The size of the array is missing! In such a case, the number of letters in the string, determines the size of the array. Hence in this case the size of the array is 6, which includes the last null character.
Programs using Array of Characters

1) Write a program that reads and stores 10 letters in an array. Print the values stored in the array.
2) Write a program that chooses randomly 20 letters from the range ‘a’ to ‘z’ and assigns them into an array. Print the array in a 2 x 10 format.

Program for (1)

```c
#include <iostream>
#include <string>

int main()
{
    char x[10];
    int i;
    
    for(i=0; i<10; i++)
    {
        cout << "Enter a letter: ";
        cin >> x[i];
    }
    
    for(i=0; i<10; i++)
    {
        cout << x[i] << " ";
    }
    cout << endl;
    return 0;
}
```

Program for (2)

```c
#include <iostream>
#include <string>

int main()
{
    char x[20];
    int i;
    srand(time(NULL));
    for(i=0; i<20; i++)
    {
        x[i] = rand()% 26 + 'a';
    }
    for(i=0; i<20; i++)
    {
        if (i%10 == 0)
            cout << endl;
        cout << x[i] << " ";
    }
    cout << endl;
    return 0;
}
```

3) Write a program that assigns into an array the first 15 terms in the sequence: ‘a’, ‘c’, ‘e’, ‘g’, ‘i’ .... Print the sequence in a 3 x 5 format.

Scrap work to determine the counters

We want to store:
- x[0] = ‘a’;
- x[1] = ‘c’;
- x[2] = ‘e’;
- x[3] = ‘g’;
- ...
- x[14] = ...

So we need a counter `i` for the array index 0, 1, 2, 3, ..., 14.

and another counter `j` for the values on the right side ‘a’, ‘c’, ‘e’ ...

For each counter note its initial value and how it increases.

Program for (3-b)

```c
#include <iostream>
#include <string>

int main()
{
    int x[15], i;
    char j;
    
    for(i=0, j= ‘a’; i<15, j=j+2)
    {
        x[i] = j;
    }
    
    for(i=0; i<15; i++)
    {
        if (i%5 == 0)
            cout << endl;
        cout << setw(2) << x[i];
    }
    cout << endl;
    return 0;
}
```
Note how similar the programs are to the ones written earlier (in page 3) for array of integers.

### Programs using Array of Characters as Strings

4) Write a program that reads and prints a user's last name as a string.
5) Write a program that reads a user's first name and last name and prints them as strings.
6) Write a program that generates a name of length 10 and prints the name as a string.
7) Write a program that stores all letters 'a' through 'z' and prints it as a string.

<table>
<thead>
<tr>
<th>Program for (4)</th>
<th>Program for (5)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define LENGTH 10</td>
<td></td>
</tr>
<tr>
<td>main()</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>char last[LENGTH];</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; “Enter your last name: ”;</td>
<td></td>
</tr>
<tr>
<td>cin &gt;&gt; last;</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; “Your last name is”</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; last</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; endl;</td>
<td></td>
</tr>
<tr>
<td>return 0;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>#define LENGTH 10</td>
<td></td>
</tr>
<tr>
<td>main()</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>char last[LENGTH], first[LENGTH];</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; “Enter your first name: ”;</td>
<td></td>
</tr>
<tr>
<td>cin &gt;&gt; first;</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; “Enter your last name: ”;</td>
<td></td>
</tr>
<tr>
<td>cin &gt;&gt; last;</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; “Your name is”</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; first &lt;&lt; “ “</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; last</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; endl;</td>
<td></td>
</tr>
<tr>
<td>return 0;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Program for (6)</th>
<th>Program for (7)</th>
</tr>
</thead>
<tbody>
<tr>
<td>#define LENGTH 11</td>
<td></td>
</tr>
<tr>
<td>main()</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>char last[LENGTH];</td>
<td></td>
</tr>
<tr>
<td>int i;</td>
<td></td>
</tr>
<tr>
<td>for(i=0; i&lt;LENGTH-1; i++)</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>last[i] = rand() % 26 + ‘a’;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>last[LENGTH-1] = ‘\0’;</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; “Name is”</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; last</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; endl;</td>
<td></td>
</tr>
<tr>
<td>return 0;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>#define LENGTH 27</td>
<td></td>
</tr>
<tr>
<td>main()</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>char Alphabet[LENGTH], j;</td>
<td></td>
</tr>
<tr>
<td>int i;</td>
<td></td>
</tr>
<tr>
<td>for(i=0,j=‘a’; i&lt;LENGTH-1; i++, j++)</td>
<td></td>
</tr>
<tr>
<td>{</td>
<td></td>
</tr>
<tr>
<td>Alphabet[i] = j;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
<tr>
<td>Alphabet[LENGTH-1] = ‘\0’;</td>
<td></td>
</tr>
<tr>
<td>cout &lt;&lt; “Alphabet ”</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; Alphabet</td>
<td></td>
</tr>
<tr>
<td>&lt;&lt; endl;</td>
<td></td>
</tr>
<tr>
<td>return 0;</td>
<td></td>
</tr>
<tr>
<td>}</td>
<td></td>
</tr>
</tbody>
</table>