Arrays creating and using Arrays.

Arrays

- It's very common to in a program to store many data values of a specified type.
    - ➔ In a sports program, you might want to store the scores for all games or the scores for each player.
    - ➔ You could write a program that does this using a different variable for each score
    - ➔ If there are a lot of games to store then this is very tedious
    - ➔ Using an array will solve this problem.
- Arrays allow you to group values together under a single name.
    - ➔ You do not separate variables for each item of data.
- 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 elements in an array have to be the same type(int, long, double, ect)
    - ➔ You cannot “mix” data types, no such thing as a single array of ints and doubles.
- Declaring to use an array in a program is similar to a normal variable that contains a single value.
    - ➔ Difference is that you need a number between square [] following the name

long number [10];

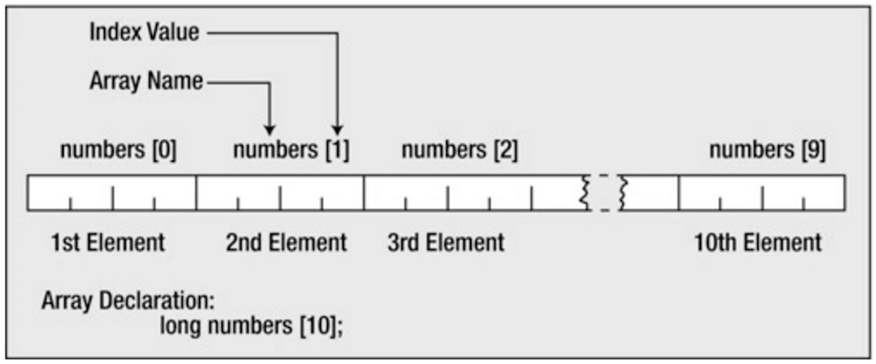
- The number between square brackets defines how many elements the array contains .
    - ➔ Called the size of the array or the array dimension.

Accessing an array’s elements

- Each of the data items stored in an array is accessed by the same name.
- You select a particular element by using an index (subscript) value between square brackets following the array name.
- Index values are sequential integers that start from zero
    - ➔ Index values for elements in an array of size 10 would be from 0-9
    - ➔ Arrays are zero based
        - ★ 0 is the index value for the first array element.
        - ★ For an array of 10 elements, index value 9 refers to the last element.
- It is a very common mistake to assume that arrays start from one.
    - ➔ Referred to as the off-by-one error
    - ➔ You can use a simple integer to explicitly reference the element that you want to access.
    - ➔ To access the fourth value in an array element, you use the expression arrayName[3].
- You 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 corresponds to one of the possible index values.

- It is very common to use a loop to access each element in an array.

```
for (i=0; i<10; ++i)
printf(" Numbers is %d", numbers[i]);
```



Array out of Bounds

- If you use an expression or a variable for an index value that is outside the range for the array, your program may crash or the array can contain garbage data .
    - ➔ Referred to as an out of bounds error.
- The compiler cannot check for out of bounds errors so your program will still compile.
- Very important to ensure that your array indexes are always within bounds.

Assigning values to an Array

- A value can be stored in an element of an array simply by specifying the array element on the left side of an equal sign.

Grades[100] =95 ;

- The value 95 is stored in element number 100 of the grades array.
- Can also use variables to assign values to an array.

Example of using an array

```
int main(void)
{
  int grades[10];                 // Array storing 10 values
  int count = 10;                 // Number of values to be read
  long sum = 0;                 // Sum of the numbers
  float average = 0.0f;       // Average of the numbers

  printf("\nEnter the 10 grades:\n");      // Prompt for the input

  // Read the ten numbers to be averaged
  for(int i = 0 ; i < count ; ++i)
  {
    printf("%2u> ",i + 1);
    scanf("%d", &grades[i]);              // Read a grade
    sum += grades[i];                     // Add it to sum
  }
```

```
 average = (float)sum/count;            // average
 printf("\nAverage of the ten grades entered is:
%.2f\n", average);

  return 0;
}
```