Arrays

An array is a collection of elements, all of the same type, stored in contiguous locations in memory. To declare an array, specify the number of elements in square brackets after the array name.

Example:

    double A [20];   // An array of 20 doubles
    char S [100];    // An array of 100 characters

All entries share the same name. Individual elements are distinguished by adding an integer to the name, called the subscript or index. The starting subscript value is zero.

Example:

    A [0] = 1.3;     // Set the first value of array A
    A [19] = -4.1;   // Set the last value of array A
Arrays

In general, a for loop is used to access elements of an array.

{
    int i;
    double Sum, Avg;

    // Set all the values of above array A to zero
    for (i = 0; i < 20; i++)
        A[i] = 0.0;

    // Compute the average of the elements in A
    Sum = 0.0;
    for (i = 0; i < 20; i++)
        Sum += A[i];
    Avg = Sum / 20;
}
Arrays

We generally picture arrays as a contiguous string of boxes:

\[ A: \]

\[ \begin{array}{cccc}
\end{array} \]

C/C++ does not check if subscripts are within bounds. \textit{E.g.}, the statement \( A [-1] = 3.1 \) will set the memory location immediately preceding \( A [0] \) to 3.1.

Do not confuse the subscript with the array value itself. The subscript is the integer between the \( [\] \)'s that identifies which array element to reference. The actual array element is the array name followed by the subscript in brackets. \textit{E.g.}, \( A [2 * i - 1] \) is an array element whose subscript is \( 2 * i - 1 \).
Arrays as Parameters

Arrays are passed to functions as pointers to the first element of the array. As a result, if a function modifies an entry in an array parameter, that modification affects the calling function. In other words, arrays are automatically reference parameters.

Example:

```c
main ()
{
    double Bob [3];
    Bob [1] = 3.14;
    Foo (Bob, 3);
    printf ("Bob [1] = %.1f\n", Bob [1]);
    return 0;
}

void Foo (double A [], int N)
{
    A [N - 2] = 5.2;
}

The output of the above is Bob [1] = 5.2.
```
More on Arrays

When declaring an array parameter, there is no need to specify the number of elements in the arrays. Since C/C++ does not check if subscripts are in bounds, it makes no difference. *E.g.*, in the preceding function `Foo` there is no number in the brackets after parameter `A`.

When declaring an array variable you *must* specify the number of elements, so the compiler will know how much memory to allocate for the array.

Arrays can be given initial values when they are declared, but putting the values in braces, separated by commas.

*Example:*

```c
```

If too few values are given, the remaining slots of the array are set to zero. When there is a list of initial values, the size of the array can be omitted—it will be determined by the number of initial values.

*Example:*

```c
int A [] = {3, 5, -2, 4, 3, 8};
```
Array Functions

double Average (double A [], int N)

// Return the average (mean) of elements
// A [0 .. (N-1)] .
// Return 0.0 is N <= 0.
{
  double Sum = 0.0;
  int i;

  if (N <= 0)
    return 0.0;

  for (i = 0; i < N; i ++)
    Sum += A [i];

  return Sum / N;
}
Array Functions

int Search (double X, double A [], int N)

// Return the subscript of the first value in
// A [0 .. (N-1)] that equals X .
// If there is none, return -1 .

{
Array Functions

void Reverse (double A [], int N)

// Reverse the order of values in
// A [0 .. (N-1)] .

{

}