Arrays

1 Creating an Array

Suppose we wanted to store 100 scores. One might try to declare 100 variables, say score0 ... score99. However, it is much better to define an array.

An array is a collection of variables of the same type. These values are accessed by an index, also called subscript. The declaration

float score[100];

creates an array with 100 values: these are score[0] through score[99]. The 0 and the 99 are the index/subscript. Note that a C array always starts at 0.

An array value can be used wherever a normal value is used. Here is code to calculate the sum of the above array:

float total = 0.0;
for(int i=0; i<100; i++) {
    total += score[i];
}

Like a normal variable, an array needs to be initialized before use. For example, one might write:

const int NUM_VALS = 20;
int count[NUM_VALS];
for(int j=0; j<NUM_VALS; j++)
    count[j] = 0;

to set all the values in the array to zero.

Occasionally, an array can be initialized at declaration:


Note that C does no range checking: you must ensure that the index is valid.

2 Sample program: arrayed.c

Here is code that reads an array from the user, then prints the array, and then determines whether every entry in the array is the same or not.
// read an array of 10 integers and check if all the same
#include <stdio.h>

int main(void)
{
    // define and get array
    const int SIZE=10;
    int A[SIZE];
    printf("Enter %d values\n", SIZE);
    int i;
    for(i=0; i<SIZE; i++)
        scanf("%d", &A[i]);

    // echo array
    printf("You entered: ");
    for(i=0; i<SIZE; i++)
        printf("%d ", A[i]);
    printf("\n");

    // check for sameNess
    int allAreSame=1;
    for(i=0; i<SIZE-1; i++) { // note where loop stops
        if( A[i]! =A[i+1] ) {
            allAreSame=0;
        }
    }
    if( allAreSame )
        printf("Same\n");
    else
        printf( "not.same\n" );

    return 0;
}

Practice Change the above program to test whether the input numbers are consecutive or not (rather than whether they are all the same).