ARRAYS

declaration syntax:	type name **[** size **] ;**
eg:	int nums[10];
pascal equivalent:	nums: array[0..9] of integer;

Note

1. Arrays are indexed from 0.
   if A is declared int A[size] then it can be indexed A[0] .. A[size-1]

2. C does no bound checking (this allows C programmers to screw up their programs in new & exciting ways).

3. When arrays are passed as args to fns they are passed by reference.

4. When passing arrays as args it is sensible (& conventional) to also pass the size (of the used part) of the array as an additional arg.

5. Fns which alter the size (of the used part) of an array should return its new size.

The next example builds 3 fns to manipulate an array of ints. The main fn calls the other fns. Algorithms on following page.

Variables

Name	Range	Type	Description
nums	series of numbers	int array	main array
Aused	nums index	int	number of used elements in nums

ReadNumbers

A	series of int	int * arg	as nums
size	+ve integer	int arg	size of A
i	A index	int	working index

CalcSum

A	series of int	int * arg	as nums
Alen	+ve integer	int arg	size of used part of A
i	A index	int	working index
sum	numeric	int	working total

Largest

A	series of int	int * arg	as nums
Alen	+ve integer	int arg	size of used part of A
i	A index	int	working index
L	numeric	int	largest value so far

main

test harness for some functions to manipulate an array of ints

```
input series of numbers
output length of series
output largest number in series output sum of numbers in series
```

ReadNumbers(A, size)

ReadNumbers reads a series of integers into an array A,

```
0 → i
while not EOF and i < size          i < size ensures i is a valid index
    input a number → A[ i ]
    i ++
return i                            i is size of used part of A
```

CalcSum(A, Alen)

Calculates & returns the sum of ints in A[0..Alen-1]

```
0 → sum
for i to Alen-1
    sum + A[ i ] → sum
return sum
```

Largest(A, Alen)

Finds & returns the largest number in A

```
A[0] → L
for i from 1 to Alen-1
    if A[ i ] > L then
        A[ i ] → L
return L
```

```
/*  P5A0.C
    this program uses functions and arrays
    programs using functions defined like this should be compiled
    using gcc
*/

#include <stdio.h>

#define ASIZE  20                   /* max size of nums array    */

int ReadNumbers( int *, int );      /* prototype for ReadNumbers */
int CalcSum( int *, int );          /* prototype for CalcSum     */
int Largest( int *, int );          /* largest number in array   */

main()
{   int nums[ ASIZE ];        /* array of integer */
    int Aused;                /* useful length of nums array */

    Aused = ReadNumbers( nums, ASIZE );
    printf( "\n %i numbers read\n", Aused );
    PrintNumbers( nums, Aused );
    printf( "\n largest number %i", Largest( nums, Aused ));
    printf( "\n sum of numbers = %i \n", CalcSum( nums, Aused ));

    return 0;
}

int ReadNumbers( int *A, int size )
    /*  reads integers into int A[size],
        returns the number of ints read before EOF
    */
{   int i, n;
    for( i=0;
         i < size && scanf( "%i", &n ) != EOF;
         i++ )
       A[i] = n;
    return i;
}

int CalcSum( int *A, int Alen )
    /*  calculates the sum of ints in int A[Alen] */
{   int sum, i;
    sum = 0;
    for( i=0; i<Alen; i++ )
        sum += A[i];

    return sum;
}

int Largest( int *A, int Alen )
    /* returns largest number in int A[Alen] */
{   int i;      /* index to A */
    int L;      /* largest number so far */
    L = A[0];
    for( i=1;  i<Alen; i++ )
        if ( A[i] > L )
            L = A[i];
    return L;
}
```

Alternative code for CalcSum

```
int CalcSum( int *A, int Alen )
    /*  calculates the sum of ints in int A[Alen] */
{   int sum, i;    for( sum = i=0; i<Alen; sum += A[i++] );

    return sum;
}
```

Alternative approach to Largest

This version uses a subsidiary fn max which returns the larger of its two integer args.

Largest(A, Alen)

```
A[0] → L
for i from 1 to Alen-1
    max( L, A[ i ] ) → L
return L
```

max(a, b)

```
if a > b
    return a
else
    return b
```

max

a	numeric	int arg	
b	numeric	int arg	

```
int max( int a, int b )
    /* returns largest of a & b */
{   return (a>b) ? a : b;}

int Largest( int *A, int Alen )
    /* returns largest number in int A[Alen] */
{   int i;      /* index to A */
    int L;      /* largest number so far */
    for( L=A[0], i=1;  i<Alen;  L = max( A[i++], L ) )
        ;
    return L;
}
```

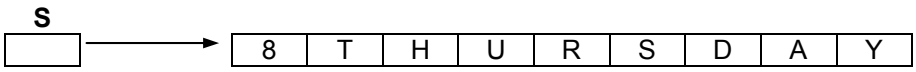
STRINGS

Two possible strategies used for representing strings in a 1 dimensional character array:

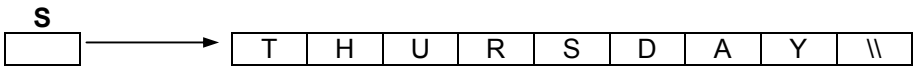
1. using explicit character count
2. using terminating symbol

using explicit character count

eg: char s[] = "THURSDAY"



using terminating symbol



C (like most modern languages) uses a terminating symbol. In C the terminator is called NULL (with a value of zero). Strings are said to be *null terminated*.

readline

Read a line of text from stdin into a character array as a null terminated string.
Reading stops at EOLN or when array is full.

readline(line, size)

```
0 → i
while i < size-1 and (input char → c) ≠ '\n'
    c → line[ i ]
    i ++
NULL → line[ i ]
return i
```

```
int readline( char *line, int size )
{   /* read a line of text into line terminated by \n */
    int i;
    char c;
    for( i=0; i < size-1 && (c = getchar()) != '\n'; )
        line[ i++ ] = c;
    line[ i ] = NULL;
    return i;
}
```