19 Pointers

In C, when an asterisk occurs in a variable declaration, *an address type* is indicated. For example,

```
char a[200], * x, y, ** z;

a is an array of characters.  Remember, its value is an address.
x is an address of a character.
y is  a character
z is an address of an address of a character.
```

In C, pointers and arrays are treated the same way. The difference is that space is reserved for an array, but not for a pointer.

If x is a pointer type, for example `int *x`, then

```
*x
```

is the value of the int stored at address x.

One can treat pointers like arrays;

```
int *x;

printf("%d\n", x[2]);  // x is not initialised so this
                       // references random parts of
                       // memory and will probably trigger
                       // a segmentation fault.
}
```

Pointer types allow what is termed 'call by reference':

```
void find_end_of_string ( char a[], int * n )
{
  for ( *n = 0; a[ *n ] != '\0'; ++ * n )
  {}
}


main()
{
   char a[] = "abc";
   int n;

   find_end_of_string ( a, &n );
   printf("find end of string returns %d\n",  n);
}
```