

## 02.4 Exploring C-pointers

### Assigning to a pointer

The function `fgets_keypad`, the source for which is shown in the introduction to Lab Exercise 02, was used in Lab Exercise 01. Recall that in `double_in` we supplied as arguments to `fgets_keypad` a character array (pointer) and its length. Instead of returning the string, the function wrote to the character array it was supplied—but remember: inside a C function arguments are assigned automatic variables. How does `fgets_keypad` assign to the array when it knows only a pointer to its first element? The secret sauce is to assign through a dereferenced pointer. Examine the source for `fgets_keypad` or consider the following example.

```
#include <stdio.h>
void foo(int * p);

int main() {
    static int x = 0;
    static int * p = &x;
    printf("before: %d\n", *p);
    foo(p);
    printf("after: %d\n", *p);
    return 0;
}

void foo(int * p) {
    *p = 3;
}
```

before: 0  
after: 3

Note that, while this sort of structure is rare among higher-level programming languages, it is quite common in C. For instance, `fgets` and `gets` have this same feature.

`char s[6] = "hello";`  
`*s = 'h'`  
`*(s+1) = 'e'`  
`*(s+2) = 'l'`

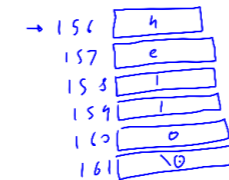
String from `vsprintf` length `n`

```
for (i=0; i<n; i++) {
    putchar_cld(string[i]);
    putchar_cld(*(string+i));
}
```

`putchar_cld(*string++);`

`char string = "hello"    n=5`  
`int x = (int) string;`  
`x = 156`

`char * s = string;`  
`for (i=0; i<n; i++) {`  
`putchar_cld(*s++);`  
`}`



i=0	s=156	'h'	s=157
i=1	s=157	'e'	s=158
i=2	s=158	'l'	s=159
i=3	s=159	'l'	s=160
i=4	s=160	'o'	s=161
i=5	i<n	s<s	x