Never apply the indirection operator to an uninitialized pointer variable. If a pointer variable p hasn't been initialized, attempting to use the value of p in any way causes undefined behavior. In the following example, the call of `printf` may print garbage, cause the program to crash, or have some other effect:

```
int *p;
printf("%d", *p);    /*** WRONG ***/
```

Assigning a value to `*p` is particularly dangerous. If p happens to contain a valid memory address, the following assignment will attempt to modify the data stored at that address:

```
int *p;
*p = 1;    /*** WRONG ***/
```

If the location modified by this assignment belongs to the program, it may behave erratically; if it belongs to the operating system, the program will most likely crash. Your compiler may issue a warning that p is uninitialized, so pay close attention to any warning messages you get.

11.3 Pointer Assignment

C allows the use of the assignment operator to copy pointers, provided that they have the same type. Suppose that `i`, `j`, `p`, and `q` have been declared as follows:

```
int i, j, *p, *q;
```

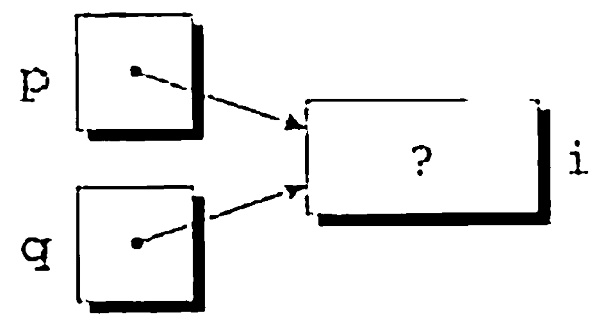
The statement

```
p = &i;
```

is an example of pointer assignment; the address of `i` is copied into p. Here's another example of pointer assignment:

```
q = p;
```

This statement copies the contents of p (the address of `i`) into q, in effect making q point to the same place as p:



Both p and q now point to `i`, so we can change `i` by assigning a new value to either `*p` or `*q`: