Consider the declaration,

```
int  i = 3 ;
```

This declaration tells the C compiler to:

(a) Reserve space in memory to hold the integer value.
(b) Associate the name **i** with this memory location.
(c) Store the value 3 at this location.

We may represent **i**'s location in memory by the following memory map.

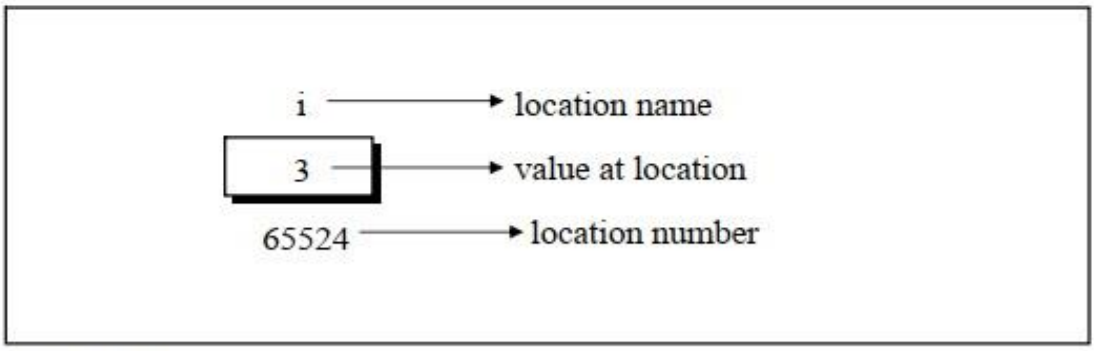

Figure 5.1

Pointers explained below in detail

Note that printing the value of ***(&i)** is same as printing the value of **i**.

The expression **&i** gives the address of the variable **i**. This address can be collected in a variable, by saying,

j = &i ;

But remember that **j** is not an ordinary variable like any other integer variable. It is a variable that contains the address of other variable (**i** in this case). Since **j** is a variable the compiler must provide it space in the memory. Once again, the following memory map would illustrate the contents of **i** and **j**.

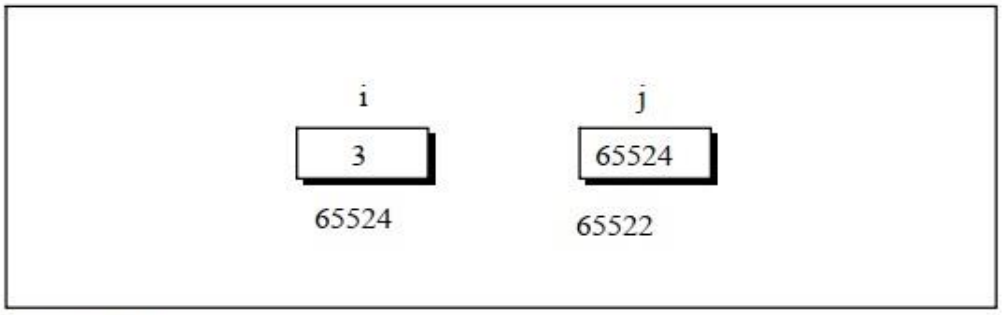

Figure 5.2

Figure 5.3 would help you in tracing out how the program prints the above output.

Remember that when you run this program the addresses that get printed might turn out to be something different than the ones shown in the figure. However, with these addresses too the relationship between **i**, **j** and **k** can be easily established.

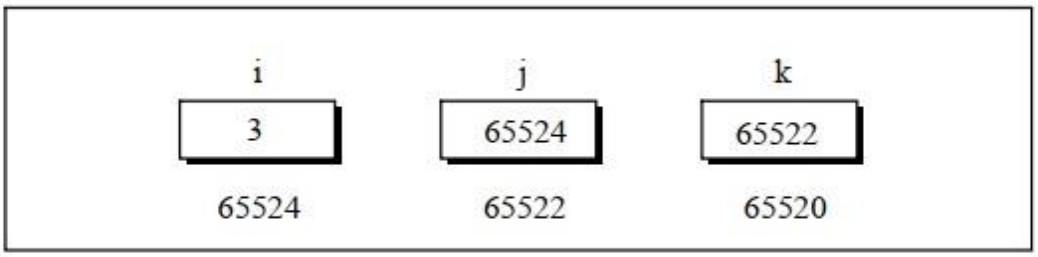

Figure 5.3