`str` represents the array into which we'll store the input, and `n` is the maximum number of characters to be read. If the input line contains more than `n` characters, `read_line` will discard the additional characters. We'll have `read_line` return the number of characters it actually stores in `str` (a number anywhere from 0 to `n`). We may not always need `read_line`'s return value, but it doesn't hurt to have it available.

getchar function ➤7.3

`read_line` consists primarily of a loop that calls `getchar` to read a character and then stores the character in `str`, provided that there's room left. The loop terminates when the new-line character is read. (Strictly speaking, we should also have the loop terminate if `getchar` should fail to read a character, but we'll ignore that complication for now.) Here's the complete definition of `read_line`:

Q&A

```
int read_line(char str[], int n)
{
  int ch, i = 0;

  while ((ch = getchar()) != '\n')
    if (i < n)
      str[i++] = ch;
  str[i] = '\0';        /* terminates string */
  return i;             /* number of characters stored */
}
```

Note that `ch` has `int` type rather than `char` type, because `getchar` returns the character that it reads as an `int` value.

Before returning, `read_line` puts a null character at the end of the string. Standard functions such as `scanf` and `gets` automatically put a null character at the end of an input string; if we're writing our own input function, however, we must take on that responsibility.

13.4 Accessing the Characters in a String

Since strings are stored as arrays, we can use subscripting to access the characters in a string. To process every character in a string `s`, for example, we can set up a loop that increments a counter `i` and selects characters via the expression `s[i]`.

Suppose that we need a function that counts the number of spaces in a string. Using array subscripting, we might write the function in the following way:

```
int count_spaces(const char s[])
{
  int count = 0, i;

  for (i = 0; s[i] != '\0'; i++)
    if (s[i] == ' ')
      count++;
  return count;
}
```