```
  cout << "The names are:\n";
  for (int i=0; i<count; i++)
    cout << "\t" << i << ". [" << name[i] << "]" << endl;
}
Enter at most 8 names with at most 23 characters:
George Washington
John Adams
Thomas Jefferson
^Z
The names are:
       0. [George Washington]
       1. [John Adams]
       2. [Thomas Jefferson]
```

Note that all the activity in the `while` loop is done within its control condition:

```
cin.getline(name[count++],20)
```

This call to the `cin.getline()` function reads the next line into `name[count]` and then increments `count`. The function returns nonzero (*i.e.*, "true") if it was successful in reading a character string into `name[count]`. When the end-of-file is signalled (with **<Control-D>** or **<Control-Z>**), the `cin.getline()` function fails, so it returns 0 which stops the `while` loop. The body of this loop is empty, indicated by the line that contains nothing but a semicolon.

A more efficient way to store C-strings is to declare an array of pointers: `char* name[4];` Here, each of the 4 components has type `char*` which means that each `name[i]` is a C-string. This declaration does not initially allocate any storage for C-string data. Instead, we need to store all the data in a buffer C-string. Then we can set each `name[i]` equal to the address of the first character of the corresponding name in the buffer. This is done in Example 8.11. This method is more efficient because each component of `name[i]` uses only as many bytes as are needed to store the C-string (plus storage for one pointer). The trade-off is that the input routine needs a sentinel to signal when the input is finished.

EXAMPLE 8.11 A String Array

This program illustrates the use of the `getline()` function with the sentinel character `'$'`. It is nearly equivalent to that in Example 8.10. It reads a sequence of names, one per line, terminated by the sentinel `'$'`. Then it prints the names which are stored in the array `name`:

```
int main()
{ char buffer[80];
  cin.getline(buffer,80,'$');
  char* name[4];
  name[0] = buffer;
  int count = 0;
  for (char* p=buffer; *p != '\0'; p++)
    if (*p == '\n')
    { *p = '\0';            //  end name[count]
      name[++count] = p+1;  //  begin next name
    }
  cout << "The names are:\n";
  for (int i=0; i<count; i++)
    cout << "\t" << i << ". [" << name[i] << "]" << endl;
}
```