```
   for(k = 1, printf("%d: Hi!\n", k); printf("k = %d\n",k),
       k*k < 26; k+=2, printf("Now k is %d\n", k) )
           printf("k is %d in the loop\n",k);
   return 0;
}
```

Programming Exercises

1. Write a program that creates an array with 26 elements and stores the 26 lowercase letters in it. Also have it show the array contents.

2. Use nested loops to produce the following pattern:

   ```
   $
   $$
   $$$
   $$$$
   $$$$$
   ```

3. Use nested loops to produce the following pattern:

   ```
   F
   FE
   FED
   FEDC
   FEDCB
   FEDCBA
   ```

   Note: If your system doesn't use ASCII or some other code that encodes letters in numeric order, you can use the following to initialize a character array to the letters of the alphabet:

   ```
   char lets[27] = "ABCDEFGHIJKLMNOPQRSTUVWXYZ";
   ```

   Then you can use the array index to select individual letters; for example, `lets[0]` is `'A'`, and so on.

4. Use nested loops to produce the following pattern:

   ```
   A
   BC
   DEF
   GHIJ
   KLMNO
   PQRSTU
   ```

If your system doesn't encode letters in numeric order, see the suggestion in programming exercise 3.

5. Have a program request the user to enter an uppercase letter. Use nested loops to produce a pyramid pattern like this:

```
    A
   ABA
  ABCBA
 ABCDCBA
ABCDEDCBA
```

The pattern should extend to the character entered. For example, the preceding pattern would result from an input value of `E`. Hint: Use an outer loop to handle the rows. Use three inner loops in a row, one to handle the spaces, one for printing letters in ascending order, and one for printing letters in descending order. If your system doesn't use ASCII or a similar system that represents letters in strict number order, see the suggestion in programming exercise 3.

6. Write a program that prints a table with each line giving an integer, its square, and its cube. Ask the user to input the lower and upper limits for the table. Use a `for` loop.

7. Write a program that reads a single word into a character array and then prints the word backward. Hint: Use `strlen()` (Chapter 4) to compute the index of the last character in the array.

8. Write a program that requests two floating-point numbers and prints the value of their difference divided by their product. Have the program loop through pairs of input values until the user enters nonnumeric input.

9. Modify exercise 8 so that it uses a function to return the value of the calculation.

10. Write a program that requests lower and upper integer limits, calculates the sum of all the integer squares from the square of the lower limit to the square of the upper limit, and displays the answer. The program should then continue to prompt for limits and display answers until the user enters an upper limit that is equal to or less than the lower limit. A sample run should look something like this:

```
Enter lower and upper integer limits: 5 9
The sums of the squares from 25 to 81 is 255
Enter next set of limits: 3 25
The sums of the squares from 9 to 625 is 5520
Enter next set of limits: 5 5
Done
```

11. Write a program that reads eight integers into an array and then prints them in reverse order.

12. Consider these two infinite series:

```
1.0 + 1.0/2.0 + 1.0/3.0 + 1.0/4.0 + ...
1.0 - 1.0/2.0 + 1.0/3.0 - 1.0/4.0 + ...
```

    Write a program that evaluates running totals of these two series up to some limit of number of terms. Hint: –1 times itself an odd number of times is –1, and –1 times itself an even number of times is 1. Have the user enter the limit interactively; let a zero or negative value terminate input. Look at the running totals after 100 terms, 1000 terms, 10,000 terms. Does either series appear to be converging to some value?

13. Write a program that creates an eight-element array of `int`s and sets the elements to the first eight powers of 2 and then prints the values. Use a `for` loop to set the values, and, for variety, use a `do while` loop to display the values.

14. Write a program that creates two eight-element arrays of `double`s and uses a loop to let the user enter values for the eight elements of the first array. Have the program set the elements of the second array to the cumulative totals of the elements of the first array. For example, the fourth element of the second array should equal the sum of the first four elements of the first array, and the fifth element of the second array should equal the sum of the first five elements of the first array. (It's possible to do this with nested loops, but by using the fact that the fifth element of the second array equals the fourth element of the second array plus the fifth element of the first array, you can avoid nesting and just use a single loop for this task.) Finally, use loops to display the contents of the two arrays, with the first array displayed on one line and with each element of the second array displayed below the corresponding element of the first array.

15. Write a program that reads in a line of input and then prints the line in reverse order. You can store the input in an array of `char`; assume that the line is no longer than 255 characters. Recall that you can use `scanf()` with the `%c` specifier to read a character at a time from input and that the newline character (`\n`) is generated when you press the Enter key.

16. Daphne invests $100 at 10% simple interest. (That is, every year, the investment earns an interest equal to 10% of the original investment.) Deirdre invests $100 at 5% interest compounded annually. (That is, interest is 5% of the current balance, including previous addition of interest.) Write a program that finds how many years it takes for the value of Deirdre's investment to exceed the value of Daphne's investment. Also show the two values at that time.

Programming Exercises

1. Write a program that reads input until encountering the # character and then reports the number of spaces read, the number of newline characters read, and the number of all other characters read.

2. Write a program that reads input until encountering #. Have the program print each input character and its ASCII decimal code. Print eight character-code pairs per line. Suggestion: Use a character count and the modulus operator (`%`) to print a newline character for every eight cycles of the loop.

3. Write a program that reads integers until 0 is entered. After input terminates, the program should report the total number of even integers (excluding the 0) entered, the average value of the even integers, the total number of odd integers entered, and the average value of the odd integers.

4. Using `if else` statements, write a program that reads input up to #, replaces each period with an exclamation mark, replaces each exclamation mark initially present with two exclamation marks, and reports at the end the number of substitutions it has made.

5. Redo exercise 4 using a `switch`.

6. Write a program that reads input up to # and reports the number of times that the sequence `ei` occurs.

> **Note**
>
> The program will have to "remember" the preceding character as well as the current character. Test it with input such as "Receive your eieio award."

7. Write a program that requests the hours worked in a week and then prints the gross pay, the taxes, and the net pay. Assume the following:

   a. Basic pay rate = $10.00/hr

   b. Overtime (in excess of 40 hours) = time and a half

   c. Tax rate: #15% of the first $300

   20% of the next $150

   25% of the rest

   Use `#define` constants, and don't worry if the example does not conform to current tax law.