

## Handout 6.5

The following program reads integer values from an input file called `input.dat` into an array. It is assumed that the file contains at most 1000 (specified by the symbolic constant `MAXSIZE`) integers. It then sorts the array into increasing order and writes the values in sorted order to the output file called `output.dat`. The sorting function uses the insertion sort algorithm.

You should study and understand the way the library functions `fscanf` and `fprintf` are used in this example.

The input and output file names are hardcoded into the program. We will see later how the program can be written so that the file names can be specified at run time.

```
#include <stdio.h>

/* Maximum number of integer values that can be sorted. */
#define MAXSIZE 1000

int main(void)
{
    char *infile = "input.dat" ; /* Input file name. */
    char *outfile = "output.dat"; /* Output file name. */

    FILE *infp; /* File pointer for input file. */
                /* It is assumed that the input file has at most */
                /* MAXSIZE integer values to be sorted.          */
    FILE *outfp; /* File pointer for output file. */

    int values[MAXSIZE]; /* Array for storing integer values. */
    int v; /* To read integers from input file. */
    int num; /* Number of input values. */
    int i; /* Temporary; loop index. */
    void insertion_sort(int [], int); /* Prototype of sorting function used. */

    /* Open the input file for reading. */
    if ((infp = fopen(infile, "r")) == NULL) { /* Open failed. */
        printf("Could not open file input.dat for reading.\n"); exit(1);
    }
    /* Open the output file for writing. */
    if ((outfp = fopen(outfile, "w")) == NULL) { /* Open failed. */
        printf("Could not open file output.dat for writing.\n"); exit(1);
    }
}
```

```

/* Both input and output files opened successfully. Read */
/* integers from input file until EOF and store them in */
/* the values array. */

num = 0;
while (fscanf(infp, "%d", &v) != EOF) { /* Note the use of fscanf. */
    values[num++] = v;
} /* End of while. */

/* Now that all values have been read in, close the input file. */
if (fclose(infp) == EOF) { /* Error in closing input file */
    printf("Error in closing file input.dat.\n");
    exit(1);
}
/* Sort the array into increasing order. */
insertion_sort(values, num);

/* Write the values in sorted order to output file, one per line. */
for (i = 0; i < num; i++)
    fprintf(outfp, "%d\n", values[i]); /* Note the use of fprintf. */

/* Close the output file. */
if (fclose(outfp) == EOF) { /* Error in closing output file */
    printf("Error in closing file output.dat.\n");
}
return 0;
} /* End of main. */

void insertion_sort(int a[], int n) {
    /* a is the array of integers to be sorted and n is the */
    /* number of entries in a. Array a is sorted in-place */
    /* into increasing order using simple insertion sort. */
    int j, k; /* Loop indices. */
    int temp; /* Temporary storage. */

    for (j = 1; j < n; j++) {
        /* Insert a[j] in the correct spot by moving others to the right. */
        temp = a[j];
        for (k = j-1; ((k >= 0) && (a[k] > temp)); k--)
            a[k+1] = a[k]; /* Body of for loop has only one statement. */
        a[k+1] = temp; /* Correct spot for the value a[j]. */
    } /* End of outer for loop. */
} /* End of insertion_sort. */

```