

7. The following two arrays represent the fixed and variable costs involved in producing each of eight items:

```
float fixed[] = { 11.31, 12.12, 13.67, 11.91, 12.30,
                  11.8, 11.00, 12.00 } ;
```

```
float variable[] = { 1.12, 1.13, 3.14, 1.35, 2.20, 1.28,
                     1.00, 2.10 } ;
```

Write a program to input an item number in the range 1 to 8 along with the number of units produced. The program should then display the cost of producing that number of units.

8. Use two **for** loops to set all the diagonal elements of a 9 by 9 integer array to 1 and all the elements not on a diagonal to 0.
9. Write a program to input values to a 4 by 5 array, search the array for values that are less than 0 and display these values along with their row and column indices.
10. Write a program to input ten integer values into an array **unsorted**. Your program should then loop through **unsorted** ten times, selecting the lowest value during each pass. For each pass through the loop, the element in **unsorted** containing the lowest value is replaced with a large value (e.g. 9999) after copying it into the next available element of another integer array **sorted**.

This is illustrated below:

```
unsorted at the start: 14 22 67 31 89 11 42 35 65 49
sorted at the start:
```

```
unsorted after the first pass: 14 22 67 31 89 9999 42 35 65 49
sorted after the first pass: 11
```

```
unsorted after the second pass: 9999 22 67 31 89 9999 42 35 65 49
sorted after the second pass: 11 14
```

etc.

Display the values in **sorted**. (Hint: see program P7C to determine the smallest value.)

11. In a magic square the rows, columns and diagonals all have the same sum. For example:

17	24	1	8	15
23	5	7	14	16
4	6	13	20	22
10	12	19	21	3
11	18	25	2	9

and

4	9	2
3	5	7
8	1	6

Write a program to read in a two-dimensional integer array and check if it is a magic square.

8.1 Variables

Every variable has a memory location.

You have a variable that is used to store data and display it.

Program 8.1

```
1 /* P8.1
2 #include <stdio.h>
3 main()
4 {
5     int i, n;
6     char ch;
7     void func1(int);
8     void func2(int);
9     printf("Enter a number: ");
10    scanf("%d", &n);
11    printf("Enter a character: ");
12    scanf("%c", &ch);
13    printf("n = %d, ch = %c\n", n, ch);
14    func1(n);
15 }
```

A sample run:

```
var1 has value 10
var2 has value 20
```

This is how the program works:

Address of variable

Do not be confused because the locations are different.