CHAPTER 11

THE SELECTION SORT ALGORITHM

Like the bubble sort algorithm covered in Chapter 11, the selection sort algorithm provides a quick and easy way to sort the items stored in a small array. The algorithm works by comparing the value stored in the first array element with the values stored in each of the remaining elements. When a smaller value is located, it is swapped with the value in the first element. At the end of the first pass, the first element contains the smallest value in the array. The algorithm then compares the value in the second element with the values in the remaining elements, swapping smaller values with the value stored in the second element. At the end of the second pass, the second element contains the next smallest value in the array. The algorithm continues comparing and swapping until the data in the array is sorted.

The code shown in WM-Figure 11-1 uses the selection sort algorithm to sort the contents of the one-dimensional `numbers` array in ascending order.

```
 1    int numbers[4] = {23, 46, 12, 35};
 2    int maxSub = 3;    //maximum subscript
 3    int temp = 0;      //used for swapping
 4
 5    for (int x = 0; x <= maxSub - 1; x += 1)
 6        for (int y = x + 1; y <= maxSub; y += 1)
 7           if (numbers[y] < numbers[x])
 8           {
 9               //swap the numbers
10               temp = numbers[y];
11               numbers[y] = numbers[x];
12               numbers[x] = temp;
13           } //end if
14        //end for
15  //end for
```

WM-Figure 11-1 Selection sort code

To help you understand the selection sort, you will desk-check the code shown in WM-Figure 11-1. Lines 1 through 3 create the `numbers` array and two variables named `maxSub` and `temp`. The `maxSub` variable stores the highest subscript in the array (3), and the `temp` variable will be used for swapping the array values (if necessary). The outer `for` clause on Line 5 creates the `x` variable and initializes it to the number 0. It also checks whether the variable's value is greater than a number that is one less than the value stored in the `maxSub` variable. The number 0 is not greater than the number 2, so the outer loop instructions are processed.

The first instruction in the outer loop is the nested `for` clause on Line 6. The clause creates the `y` variable and initializes it to a number that is one more than the value stored in the `x` variable; in this case, it initializes the `y` variable to 1. The clause also checks whether the `y` variable's value (1) is greater than the value stored in the `maxSub` variable (3). The number 1 is not greater than the number 3, so the nested loop instructions are processed. See the desk-check table in WM-Figure 11-2.

numbers[0]	numbers[1]	numbers[2]	numbers[3]
23	46	12	35

maxSub	temp	x	y
3	0	0	1

WM-Figure 11-2 Desk-check table before Line 7 is processed

The first instruction in the nested loop is the `if` clause on Line 7. The clause's condition compares the value stored in the `numbers[1]` element with the value stored in the `numbers[0]` element. The condition evaluates to false because 46 is not less than 23. Therefore, the computer skips over the instructions in the selection structure's true path and then returns to the beginning of the nested loop on Line 6. The `for` clause on Line 6 adds 1 to the contents of the `y` variable, giving 2. It also checks whether the `y` variable's value is greater than the value stored in the `maxSub` variable. The number 2 is not greater than the number 3, so the nested loop instructions are processed again.

The first instruction in the nested loop is the `if` clause on Line 7. This time, the clause's condition compares the value stored in the `numbers[2]` element with the value stored in the `numbers[0]` element. The condition evaluates to true because 12 is less than 23. As a result, the instructions in the selection structure's true path are processed. Those instructions swap the values stored in the `numbers[2]` and `numbers[0]` elements, as indicated in WM-Figure 11-3.

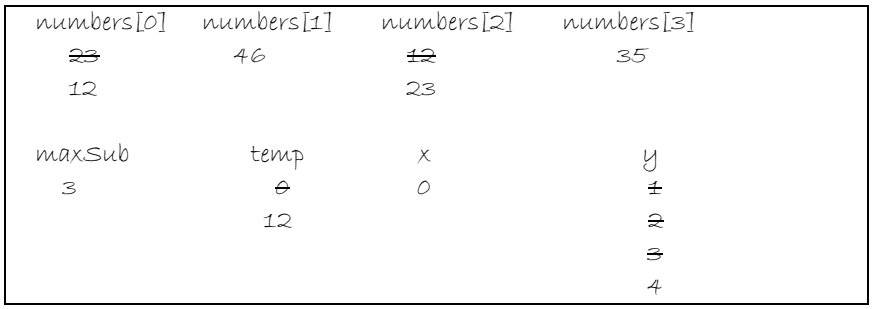
numbers[0]	numbers[1]	numbers[2]	numbers[3]
~~23~~	46	~~12~~	35
12		23	

maxSub	temp	x	y
3	0	0	~~1~~
			2

WM-Figure 11-3 Desk-check table after the first swap during the first pass

The nested `for` clause is processed next. The clause adds 1 to the contents of the `y` variable, giving 3. It also checks whether the `y` variable's value is greater than the value stored in the `maxSub` variable. The number 3 is not greater than the number 3, so the nested loop instructions are processed again.

The first instruction in the nested loop is the `if` clause on Line 7. The clause's condition compares the value stored in the `numbers[3]` element with the value stored in the `numbers[0]` element. The condition evaluates to false because 35 is not less than 12. As a result, the instructions in the selection structure's true path are skipped over and processing continues once again with the nested `for` clause on Line 6. The clause adds 1 to the contents of the `y` variable, giving 4. It also checks whether the `y` variable's value is greater than the value stored in the `maxSub` variable. The number 4 is greater than the number 3, so the nested loop ends. WM-Figure 11-4 shows the status of the desk-check table at the end of the first pass.



WM-Figure 11-4 Desk-check table at the end of the first pass

The outer loop's `for` clause on Line 5 is processed next. The clause adds 1 to the contents of the `x` variable, giving 1. It also checks whether the variable's value is greater than a number that is one less than the value stored in the `maxSub` variable. The number 1 is not greater than the number 2, so the outer loop instructions are processed again.

The first instruction in the outer loop is the nested `for` clause on Line 6. The clause creates the `y` variable and initializes it to a number that is one more than the value stored in the `x` variable; in this case, it initializes the `y` variable to 2. The clause also checks whether the `y` variable's value is greater than the value stored in the `maxSub` variable. The number 2 is not greater than the number 3, so the nested loop instructions are processed.

The first instruction in the nested loop is the `if` clause on Line 7. The clause's condition compares the value stored in the `numbers[2]` element with the value stored in the `numbers[1]` element. The condition evaluates to true because 23 is less than 46. As a result, the instructions in the selection structure's true path are processed. Those instructions swap the values stored in the `numbers[2]` and `numbers[1]` elements, as indicated in WM-Figure 11-5.

numbers[0]	numbers[1]	numbers[2]	numbers[3]
~~23~~	~~46~~	~~12~~	35
12	23	~~23~~	
		46	

maxSub	temp	x	y
3	~~0~~	~~0~~	~~1~~
	~~12~~	1	~~2~~
	23		~~3~~
			~~4~~
			2

WM-Figure 11-5 Desk-check table after the first swap during the second pass

The nested `for` clause on Line 6 is processed next. The clause adds 1 to the contents of the `y` variable, giving 3. It also checks whether the `y` variable's value is greater than the value stored in the `maxSub` variable. The number 3 is not greater than the number 3, so the nested loop instructions are processed again.

The first instruction in the nested loop is the `if` clause on Line 7. The clause's condition compares the value stored in the `numbers[3]` element with the value stored in the `numbers[1]` element. The condition evaluates to false because 35 is not less than 23. As a result, the instructions in the selection structure's true path are skipped over and processing continues with the nested `for` clause on Line 6. The clause adds 1 to the contents of the `y` variable, giving 4. It also checks whether the `y` variable's value is greater than the value stored in the `maxSub` variable. The number 4 is greater than the number 3, so the nested loop ends. WM-Figure 11-6 shows the status of the desk-check table at the end of the second pass.

numbers[0]	numbers[1]	numbers[2]	numbers[3]
~~23~~	~~46~~	~~12~~	35
12	23	~~23~~	
		46	

maxSub	temp	x	y
3	~~0~~	~~0~~	~~1~~
	~~12~~	1	~~2~~
	23		~~3~~
			~~4~~
			~~2~~
			~~3~~
			4

WM-Figure 11-6 Desk-check table at the end of the second pass

The outer `for` clause on Line 5 is processed next. The clause adds 1 to the contents of the `x` variable, giving 2. It also checks whether the variable's value is greater than a number that is one less than the value stored in the

maxSub variable. The number 2 is not greater than the number 2, so the outer loop instructions are processed again.

The first instruction in the outer loop is the nested for clause on Line 6. The clause creates the y variable and initializes it to a number that is one more than the value stored in the x variable; in this case, it initializes the y variable to 3. The clause also checks whether the y variable's value is greater than the value stored in the maxSub variable. The number 3 is not greater than the number 3, so the nested loop instructions are processed.

The first instruction in the nested loop is the if clause on Line 7. The clause's condition compares the value stored in the numbers[3] element with the value stored in the numbers[2] element. The condition evaluates to true because 35 is less than 46. As a result, the instructions in the selection structure's true path are processed. Those instructions swap the values stored in the numbers[3] and numbers[2] elements.

The nested for clause on Line 6 is processed next. The clause adds 1 to the contents of the y variable, giving 4. It also checks whether the y variable's value is greater than the value stored in the maxSub variable. The number 4 is greater than the number 3, so the nested loop ends and processing continues with the outer for clause on Line 5. The clause adds 1 to the contents of the x variable, giving 3. It also checks whether the variable's value is greater than a number that is one less than the value stored in the maxSub variable. The number 3 is greater than the number 2, so the outer loop ends. The completed desk-check table is shown in WM-Figure 11-7. Notice that the array values are now sorted in ascending order.

numbers[0]	numbers[1]	numbers[2]	numbers[3]
~~23~~	~~46~~	~~12~~	~~35~~
12	23	~~23~~	46
		~~46~~	
		35	

maxSub	temp	x	y
3	~~0~~	~~0~~	~~1~~
	~~12~~	~~1~~	~~2~~
	~~23~~	~~2~~	~~3~~
	35	3	~~4~~
			~~2~~
			~~3~~
			~~4~~
			~~3~~
			4

WM-Figure 11-7 Completed desk-check table