Laboratory 06 – arrays

10.11.2020

1. [4points] Write a C program to read elements in an array and find the sum of array elements.
a) Declare an array of integers of size `10`. In your program, use the preprocessor `#define` directive to create an `SIZE` constant that has the value `10`.
b) Use a `for` loop and `scanf` functions to load values into an array.
c) Use a `for` loop to calculate the sum of all elements in an array. An additional variable is needed to calculate the sum.
d) Print all values from the array to the screen.
e) Print all values from the array to the screen in the reverse order.
f) Print the sum of the elements from an array.

Each loop has its own local index.
What is the minimum number of loops needed?

Test data:

```
Enter 10 elements in the array : 1 23 4 56 67 89 12 45 56 78

Elements in array are:
arr[0] = 1
arr[1] = 23
arr[2] = 4
arr[3] = 56
arr[4] = 67
arr[5] = 89
arr[6] = 12
arr[7] = 45
arr[8] = 56
arr[9] = 78

Array in reverse order:
arr[9] = 78
arr[8] = 56
arr[7] = 45
arr[6] = 12
arr[5] = 89
arr[4] = 67
arr[3] = 56
arr[2] = 4
arr[1] = 23
arr[0] = 1

Sum of all elements of array = 431
```

2. [4points] Write a C program to input elements in an array from user, find maximum and minimum element in array.

a) Declare an array of integers of size `10`. In your program, use the preprocessor `#define` directive to create an `SIZE` constant that has the value `10`.
b) Use a `for` loop and `scanf` functions to load values into an array.

c) Declare two variables `max` and `min` to store maximum and minimum. Assume that the first element in the array is both the maximum and the minimum.
d) Iterate through the array to find the maximum and minimum element in the array. Run a loop from the first to the last element of the array.
e) Print the maximum and minimum element.

Each loop has its own local index.
What is the minimum number of loops needed?

Test data:

```
Enter 10 elements in the array : 23 4 56 67 89 1 12 45 56 78

Elements in array are:
arr[0] = 23
arr[1] = 4
arr[2] = 56
arr[3] = 67
arr[4] = 89
arr[5] = 1
arr[6] = 12
arr[7] = 45
arr[8] = 56
arr[9] = 78
Maximum element = 89
Minimum element = 1
```

3. [5points] Write a C program that will generate arrays of 20 integer pseudo-random numbers in the range given by the user. Then the program generates one pseudo-random number and checks if that number is in the array.

Use the `rand` function from the <`stdlib.h`> header to generate pseudo-random numbers.

a) Declare an array of integers of size `20`. In your program, use the preprocessor `#define` directive to create an `SIZE` constant that has the value `20`.
b) Declare two variables, the start and end of the pseudo-random number generation range. Load values with `scanf`.
c) Fill the array with pseudo-random values in the range given by the user.
d) Generate the value we'll look for in the array.
e) Declare a `flag` that will mean that a value has been found in the array. Initialize the `flag` with a value that means the number you are looking for was not found.
f) Use the `for` loop to check if the value you are looking for is in the array. If you find the value, change the `flag` value and `break` the loop.
g) If the `flag` indicates that a value has been found, print the value and its index in the array.
h) Otherwise, print a message that the value has not been found.

Test data:

```
Enter the limits of the number generation range:
1 23
```

```
Elements in array are:
arr[0] = 12
arr[1] = 1
arr[2] = 7
arr[3] = 3
arr[4] = 7
arr[5] = 1
arr[6] = 15
arr[7] = 2
arr[8] = 12
arr[9] = 2
arr[10] = 18
arr[11] = 17
arr[12] = 7
arr[13] = 18
arr[14] = 10
arr[15] = 18
arr[16] = 16
arr[17] = 11
arr[18] = 9
arr[19] = 13
17 is found at position 11

Enter the limits of the number generation range:
2 123

Elements in array are:
arr[0] = 31
arr[1] = 56
arr[2] = 5
arr[3] = 117
arr[4] = 9
arr[5] = 51
arr[6] = 110
arr[7] = 122
arr[8] = 91
arr[9] = 25
arr[10] = 8
arr[11] = 49
arr[12] = 104
arr[13] = 83
arr[14] = 11
arr[15] = 42
arr[16] = 54
arr[17] = 82
arr[18] = 26
arr[19] = 52
119 is not found in the array
```

4. [7points] Write a C program that implements selection sort.

The idea of the selection sort algorithm:

```
Find the smallest element. Swap it with the first element.

Find the second-smallest element. Swap it with the second element.

Find the third-smallest element. Swap it with the third element.

Repeat finding the next-smallest element, and swapping it into the
correct position until the array is sorted.
```

Example of the selection sort algorithm:

```
arr[] = 64 25 12 22 11

First iteration:
// Find the minimum element in arr[0...4]
// and place it at beginning
11 25 12 22 64

Second iteration:
// Find the minimum element in arr[1...4]
// and place it at beginning of arr[1...4]
11 12 25 22 64

Third iteration:
// Find the minimum element in arr[2...4]
// and place it at beginning of arr[2...4]
11 12 22 25 64

Fourth iteration:
// Find the minimum element in arr[3...4]
// and place it at beginning of arr[3...4]
11 12 22 25 64
```

a)) Declare an array of integers of size `10`. In your program, use the preprocessor `#define` directive to create an `SIZE` constant that has the value `10`.
b) Fill the array with pseudo-random values, from 0 to 100
c) Print the array before sorting.
d) Two nested loops are needed to implement sorting. In order for the outer loop to place the array elements in ascending order, the inner loop must look for the minimum value in the part of the array that has not yet been sorted. When the inner loop finds the minimum, the outer loop must put the smallest element at the beginning of the area that has not yet been sorted.
e) Analyze the example of the algorithm above to determine the starting and ending indexes for both loops. How does the number of iterations in the outer loop depend on the size of the array? How does the starting value of the inner loop index change in subsequent iterations?
f) Print the array after sorting.

Test data:

```
Elements in array are:
arr[0] = 32
arr[1] = 32
arr[2] = 54
arr[3] = 12
arr[4] = 52
arr[5] = 56
arr[6] = 8
arr[7] = 30
arr[8] = 44
arr[9] = 94

Elements in sorted array are:
arr[0] = 8
arr[1] = 12
arr[2] = 30
arr[3] = 32
arr[4] = 32
arr[5] = 44
arr[6] = 52
arr[7] = 54
arr[8] = 56
arr[9] = 94
```

Next time:
Laboratory 07 – Functions