Lebanese University
Faculty of Science
BS Computer Science
2nd Year – S3

I2204 - Imperative Programming

Dr Siba Haidar

Lebanese University
Faculty of Science
BS Computer Science
2nd Year – S3

Pointers and Arrays

Chapter 2

Exercises

Exercise: 3 pointers

write a program which

- declares
  - an integer x and a pointer to int px
  - a double y=13.4 and a pointer to double py
  - a char z and a pointer to char pz

- initilizes the value so that the primitive variabes have values and the pointers point each to the corresponding variable
- diplays the values of the pointers, of their pointees and the addresses of x, y and z

- compile, run and test your program
- **draw the memory state of your program**

Exercise: 3 pointers - Revisited

- in exercise "Exercise: 3 pointers” what do we get if we add the following instruction
  - px = &y;
  - printf("the content of px is %d", *px);

- **apply the changes on the memory state for these 2 instructions**

Exercise: inc5

- Write a function "inc5" which increments by 5 a variable through its address
- We give the function incTest below

```
void inc5Test(){
     int i=7;
     printf("before: i = %d\n",i);//must print 7
     inc5(&i);
     printf("after: i = %d\n",i);//must print 12
}
```

- compile, run and test your program
- **draw the memory state of your program**

Exercise: swap

- Write a function ”swap" which swaps the values of two variables, given their addresses
- We give the function swapTest below

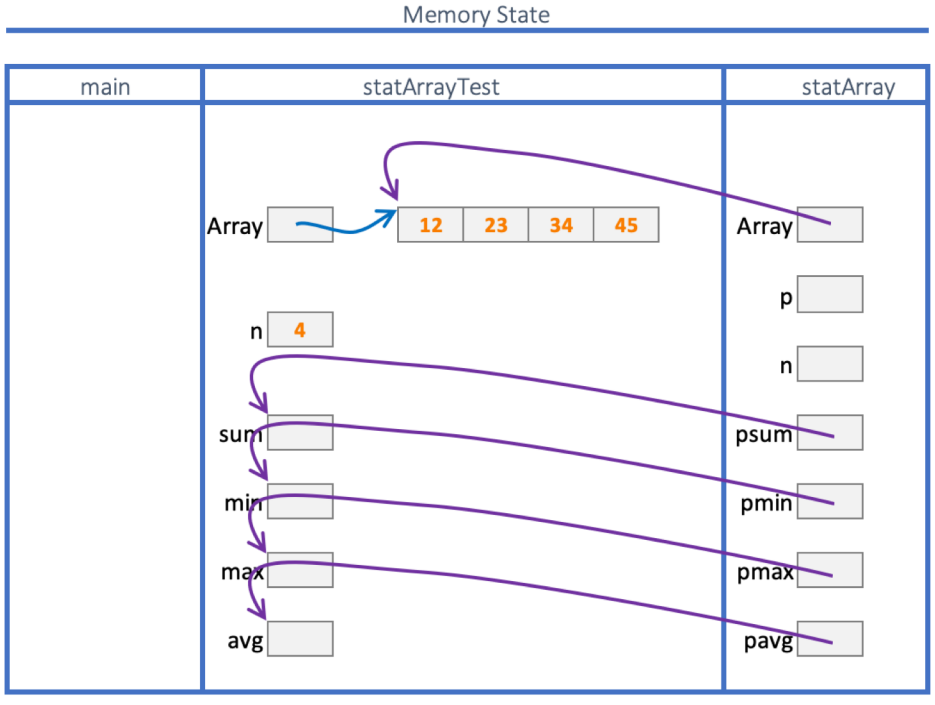
```
void swapTest(){
      int i=7, j=56;
      printf("before : i = %d, j = %d\n",i,j);
      swap(&i, &j);
      printf("before : i = %d, j = %d\n",i,j);
}
```

- compile, run and test your program
- **draw the memory state of your program**

Exercise: statArray

- write a function statArray which, given an array of integers, calculates and returns four aggregation values
  - its sum
  - its average
  - its maximum value
  - its minimum value
- **HINT:**
  - since the function can only return one value you should use pointers so that the function statArray fill the pointees in the memory of the caller
- write the test function statArrayTest
  - use a big array

- compile, run and test your program
- **draw the memory state of your program**



function call

params in statArray		args in statArrayTest
int * Array	=	Array
int n	=	n
int *psum	=	&sum
float* pavg	=	&avg
int * pmin	=	&min
int *pmax	=	&max

Exercise: recStatArray

- Write a **RECURSIVE** function recStatArray which does the same thing as in exercise 5.
  - pay attention for the average (you need a helper function to add a counter parameter)

- compile, run and test your program
- **draw the memory state of your program**

Big Array Sample

min =	24.00
max =	2,996.00
sum =	434,959.00
avg =	1,449.86

692, 494, 953, 2359, 2819, 2402, 151, 1465, 1259, 2651, 1260, 186, 733, 1012, 287, 928, 2802, 2231, 2013, 2370, 2925, 1473, 1791, 1168, 2353, 1942, 2706, 2457, 350, 1006, 998, 2509, 760, 2190, 738, 1510, 578, 1087, 2562, 1975, 2988, 442, 994, 1380, 1248, 878, 1485, 570, 172, 1689, 1939, 2216, 2047, 1337, 882, 1975, 1081, 1956, 44, 1236, 743, 815, 896, 2845, 1165, 2035, 2053, 1174, 1410, 229, 1693, 2145, 660, 1765, 2442, 1474, 1210, 2930, 2660, 1882, 2636, 1501, 876, 833, 387, 447, 2304, 2621, 98, 2135, 586, 1835, 1135, 1522, 1169, 1228, 344, 767, 317, 2423, 2577, 2187, 2296, 1593, 758, 328, 1379, 1736, 2699, 364, 1809, 914, 953, 999, 632, 775, 915, 163, 134, 585, 1973, 1781, 229, 999, 2315, 885, 1877, 1494, 2996, 2932, 163, 2411, 137, 1437, 1953, 2643, 2186, 863, 50, 1848, 1228, 2592, 473, 2857, 2788, 2316, 868, 133, 1495, 261, 1300, 2464, 1291, 1949, 375, 1953, 1595, 2199, 1562, 2677, 152, 1517, 1020, 1907, 2879, 2822, 1574, 2392, 450, 315, 1006, 2944, 430, 1775, 1698, 2517, 481, 2052, 346, 2989, 2197, 2467, 1882, 1812, 1610, 2621, 1034, 1179, 1293, 2119, 2791, 2666, 1686, 2456, 2261, 1217, 1261, 1201, 2169, 2935, 1039, 754, 2632, 2250, 108, 1107, 578, 1715, 208, 1357, 1052, 1961, 1958, 400, 1261, 1137, 1699, 233, 473, 155, 587, 904, 1438, 1092, 1857, 2492, 1914, 1066, 638, 1019, 362, 1767, 1973, 321, 1121, 2488, 1242, 1796, 1487, 24, 34, 2277, 799, 1535, 2365, 1155, 134, 2594, 629, 630, 600, 2786, 115, 2720, 2989, 142, 1367, 24, 2485, 2583, 1324, 1022, 2097, 436, 177, 925, 1124, 2614, 694, 2548, 1520, 1942, 2334, 1681, 2526, 1676, 2111, 1855, 80, 2661, 654, 1292, 280, 1484, 2484, 835, 1888, 1193, 395, 1425, 2518, 610, 1780, 2067, 51, 467, 1409, 1710, 1448, 433

Exercises: To Do

- Solve old exercises using pointer syntax:

1. printArray (iterative & recursive)
2. statArray (iterative & recursive)
3. quicksort
4. checkSorted seen in lab (hackerRank), then use it to check whether your quicksort works fine
5. binarySearch
6. traditionalSearch
7. write a final program to read the given sample big array, quicksort it, then choose a random value from within this array, and compare both searches in terms of number of operations needed to find this value.