Department of Information Systems and Technologies

CTIS 152 – Algorithms and Data Structure

FALL 2024 - 2025

Lab Guide #2 – Week 3 – 1

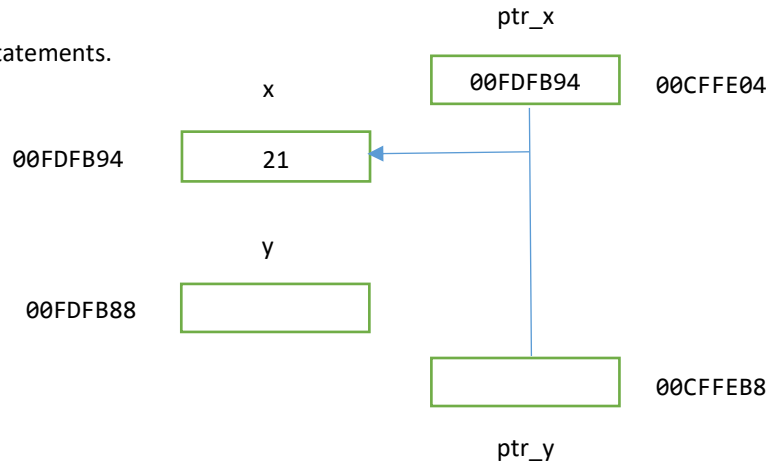
OBJECTIVE : Pointer Operations and Pointers as Function Parameters

Instructor : Burcu LİMAN
Assistants : Berk ÖNDER, Engin Zafer KIRAÇBEDEL

Use **pointer notation** instead of subscript notation!

Q1. a) Suppose that a C program segment contains the following statements.

```
int x = 21, y;
 int *ptr_x = &x;
int *ptr_y;
*ptr_x = x + 2;
y = *ptr_x + 5;
ptr_y = ptr_x;
*ptr_x = x + y;
```



Write a C program including the above program segment in order to find the final values of:

- &x
- &y
- ptr_x
- *ptr_x
- x
- ptr_y
- *ptr_y
- (ptr_x + 3)
- (*ptr_x + 3)

Project Name: LG2_Q1a
File Name: Q1a.cpp

b) Suppose that a C program contains the following statement:

```
int arr[10] = { 35, 28, 6, 1, 66, 81, 19, 34, 99 };
```

According to this statement, examine the values of:

```
arr
arr+2
*(arr+2)
*arr+4
arr+5
*(arr+7)
```

Check the results by the program.

Project Name: LG2_Q1b
File Name: Q1b.cpp

Q2. Write a C program that will initialize a one-dimensional integer array of size 6 with the numbers like 6, 36, 216, 1296 … Whence completed, the program should also give an output of these values to the text file **"sequence.txt"** in the given format below, including their addresses and pointer iterations.

Example Run:

```
Successful! Please see the sequence.txt file for the output.
```

Content of the sequence.txt file

```
Element Name      Value   Address
-------------     -----   --------
*(nums +   0)         6   006FFBA4
*(nums +   1)        36   006FFBA8
*(nums +   2)       216   006FFBAC
*(nums +   3)      1296   006FFBB0
*(nums +   4)      7776   006FFBB4
*(nums +   5)     46656   006FFBB8
```

Project Name: LG2_Q2
File Name: Q2.cpp

Q3. Write a C program that reads an array of n integers, where n is max 250, from the user as input data, until a sentinel value –99 is entered. Your program should create a text file as **"matchedItems.txt"** for storing duplicate elements. Write the size of these elements as in the example run.

NOTE: The user can enter the same number at most twice! Don't need to validate it ;)

Example Run:

```
Enter a number: 13
Enter a number: 21
Enter a number: 36
Enter a number: 13
Enter a number: 6
Enter a number: 99
Enter a number: 78
Enter a number: 6
Enter a number: -99

Total number of duplicate elements found in the array -> 2
```

matchedItems.txt

```
13
6
```

Project Name: LG2_Q3
File Name: Q3.cpp

Q4. Write a C program that reads one integer list from the text file named **"numbers.txt"**, finds and displays the minimum and the maximum number in the list. Then, it calculates and displays the average of these numbers excluding the minimum and the maximum value.

Write the following functions;

- **readList :** reads a list of numbers from the text file and returns the size of the list.
- **findMinMax :** finds the minimum and the maximum value in the list and returns them.
- **findAvgExcMinMax :** calculates the average of the numbers in the list excluding the minimum and maximum value and returns the result.

Example Run:

```
Min value : 6
Max value : 93

Average excluding the min and max value : 46.25
```

numbers.txt

```
78 93 81 28 33 6 9 15 35 91
```

Project Name: LG2_Q4
File Name: Q4.cpp

Additional Questions

AQ1. Write a C program that reads a list of integers from a text file named **"numbersAQ1.txt"** into an array, displays the array content and writes the perfect numbers into the **"perfect.txt"** (Using the functions below), displays the file content and the number of perfect numbers.

Write the following functions;

- **readFromFile:** takes the file pointer and the array as parameters, reads the numbers from the file into the specified array and returns the actual number of elements.
- **display:** takes the array and its size as input parameters and displays the content of the array.
- **isPerfect**: decides whether an integer is perfect or not. **Perfect number is a number where, sum of its divisors (except itself) is equal to that number.** For example; 28 is perfect number because, it is divisors are 1,2,4,7,14 and sum of these numbers is equal to 28.
- **findPerfect:** takes the array and its size as input parameters, finds the perfect numbers in the array and stores them in another array. The function should return the number of perfects.
- **writeToFile:** takes the output file pointer and the array which keeps the perfect numbers and its size as parameters, writes the perfect numbers into the file.

a. Code the program by using subscript notation

b. Code the program using pointer notation

numbersAQ1.txt

```
7 496 25 77 28 13 17 19 21 55 38 43 82 41 35 12 6 82
```

perfect.txt

```
496   28    6
```

Example Run:

```
The list of numbers in the file:
  7 496  25  77  28  13  17  19  21  55  38  43  82  41  35  12   6  82
There are 18 numbers in the array, 3 of them are perfect numbers
```

Project Name: LG2_AQ1
File Name: AQ1.cpp

AQ2. A sparse matrix is a two-dimensional array in which a large number of the elements are zero.

Write a C program that interactively creates such a matrix. Your program should get the values, such as array dimensions, number of nonzero elements, as well as whichever value of each of these rows and columns of the array should be a nonzero value, using the function **getNonzeroElement**.

```
void getNonzeroElement (rowSize,   // number of rows in the matrix
                        colSize,   // number of columns in the matrix
                        *rowIndex, // row index to put the value in
                        *colIndex, // column index to put the value in
                        *value)    // value
```

This function inputs the size of the matrix as parameters, then reads a rowIndex, a colIndex, and a value from the user, if these indices are within the matrix boundaries, returns these indices and the value as output parameters. If invalid asks the user for new data. Don't forget to avoid subscript notation (i.e., use pointers to move in the array).

NOTE: Don't forget to check whether the two-dimensional array is a sparse matrix or not by using the following criteria;

```
numNonzero > rows * cols / 4
```

Example Run:

```
Enter number of rows and columns: 4 4

Enter number of nonzero elements: 8
Enter number of nonzero elements: 3

Enter row and column indexes and nonzero elements: 0 8 9
Enter row and column indexes and nonzero elements: 0 1 5
Enter row and column indexes and nonzero elements: 1 2 8
Enter row and column indexes and nonzero elements: 3 3 1

The Sparse Matrix:

   0   5   0   0
   0   0   8   0
   0   0   0   0
   0   0   0   1
```

Project Name: LG2_AQ2
File Name: AQ2.cpp