Department of Information Systems and Technologies

CTIS151 – Introduction to Programming

FALL 2024- 2025

Lab Guide #15

OBJECTIVES : One-Dim Arrays input + output parameters

Instructor : Serpil TIN
Assistants : Berk ÖNDER, Efe Mert ŞAHİNKOÇ, Hatice Zehra YILMAZ

Q1. Write a C program that reads book information (unique book IDs, ratings, and genres) from the file **books.txt** into three parallel arrays with a maximum size of 30. The program should display the information for all books, and then prompt the user to enter a book ID to search for. If the book is found, the program displays its information; otherwise, it shows an appropriate message.

Write the following functions;

- **`readFromFile:`** takes the file pointer and three parallel arrays (book IDs array, ratings array, genres array) as parameters. Reads the book ID, rating, and genre information from the file into the three arrays. The function returns the number of books.
- **`displayBook:`** takes the book ID, rating, and genre as input parameters and displays the information as shown in the example run. (Genres: F - Fiction, N - Non-Fiction, M - Mystery)
- **`display:`** takes the IDs array, ratings array, genres array, and the number of books as input parameters. Displays the information for all books as shown in the example run by calling the function **displayBook.**
- **`searchBook:`** takes the one-dimensional book ID array, its size, and a book ID to search for. Searches for the book ID in the array. If the book ID exists, the function returns its position; otherwise, it returns -1.

Project Name: LG15_Q1
File Name: Q1.cpp

Example Run:

```
There are 11 books in the array

Book ID  Rate   Genre
*******  ****   ******
   1234   8.9   Fiction
   1256   7.5   Mystery
   1489   9.2   Non-Fiction
   1785   6.8   Fiction
   4785   8.0   Mystery
   3256   9.5   Non-Fiction
   2564   6.7   Fiction
   7895   8.4   Mystery
   3654   7.1   Fiction
   5874   9.0   Non-Fiction
   6852   9.3   Mystery

Enter a book ID to search (negative ID to stop): 9854
The searched book ID 9854 is NOT found

Enter a book ID to search (negative ID to stop): 1785
   1785   6.8   Fiction

Enter a book ID to search (negative ID to stop): 6852
   6852   9.3   Mystery

Enter a book ID to search (negative ID to stop): -1
```

books.txt

```
1234 8.9 F
1256 7.5 M
1489 9.2 N
1785 6.8 F
4785 8.0 M
3256 9.5 N
2564 6.7 F
7895 8.4 M
3654 7.1 F
5874 9.0 N
6852 9.3 M
```

Q2. Write a program that reads information about a group of employees from the text file **employees.txt** and stores the data in four arrays. The data structure within the file is column-based: **Employee ID, Gender, Age,** and **Salary.**

The program displays a menu and then performs one of the following tasks based on the user's choice: displays the list of employees with their salaries, calculates average salaries, or searches for specific employee information.

Write the following functions;

- **`dispMenu`:** displays a menu with the following items, and then, asks for the user to input their choice, whilst also doing validation where necessary:

```
1. Display Salaries
2. Display Average Salaries
3. Display Employee Info
4. Exit
```

- **`displaySalaries`:** Displays the Employee IDs along with their salaries.
- **`displayAverages`:** Displays the average salary for male employees, female employees, and all employees.
- **`findEmployee`:** Searches for a specific employee by ID and returns the index of the employee. If the ID is not found, it returns -1.

Project Name: LG15_Q2
File Name: Q2.cpp

employees.txt

```
1001 M 28 5000
1002 F 25 4500
1003 M 30 5200
1004 F 27 4700
1005 M 35 5100
1006 F 29 4800
1007 M 40 6000
1008 F 22 4300
1009 M 32 5500
1010 F 26 4700
```

Example Run:

```
1. Display Salaries
2. Display Average Salaries
3. Display Employee Info
4. Exit
Enter your choice: 5
Wrong choice! Enter again: 0
Wrong choice! Enter again: 1
Employee ID   Salary
***********   ******
       1001     5000
       1002     4500
       1003     5200
       1004     4700
       1005     5100
       1006     4800
       1007     6000
       1008     4300
       1009     5500
       1010     4700

1. Display Salaries
2. Display Average Salaries
3. Display Employee Info
4. Exit
Enter your choice: 2
Average salary for male employees: 5360
Average salary for female employees: 4600
Average salary for all employees: 4980

1. Display Salaries
2. Display Average Salaries
3. Display Employee Info
4. Exit
Enter your choice: 3
Enter Employee ID: 2000
Employee not found!

1. Display Salaries
2. Display Average Salaries
3. Display Employee Info
4. Exit
Enter your choice: 3
Enter Employee ID: 1003
Employee Info:
ID     : 1003
Gender : M
Age    : 30
Salary : 5200

1. Display Salaries
2. Display Average Salaries
3. Display Employee Info
4. Exit
Enter your choice: 4
```

Q3. Write a C program that reads words from **words.txt,** finds the words that are palindromes, and writes them into **palindromes.txt.**

Write the following functions;

- **`isPalindrome:`** takes a character array and the word length as input parameters, and returns whether the word is a palindrome or not.
- **`writeToFile:`** takes a file pointer, a character array, and the word length as input parameters, and writes the word into a file.

<u>**Hint:**</u> Palindromes are a sequence of characters that are the same when read forward or backward. E.g. “mom”, “madam”, “radar”, and “refer” are palindrome words.

`words.txt`

```
level night ball racecar hannah start defied computer mark nun madam dog soon radar kayak
table deed apple wow moon
```

`palindromes.txt`

```
level racecar hannah nun madam radar kayak deed wow
```

Project Name: LG15_Q3
File Name: Q3.cpp

Q4. Write a C program that reads a cipher key from **key.txt** into an array, reads and deciphers the sentence from **ciphered.txt** according to the cipher key, and writes the deciphered text into **deciphered.txt.**

Write the following functions;

- **`readFromFile:`** takes a file pointer as an input parameter, reads the content from the text file, stores it in an array, and returns it.
- **`findLetterIndex:`** takes the key array and a character as input parameters, finds, and returns the index of the character in the key array.
- **`decipherChar:`** takes a file pointer, the key array, and a character as input parameters, finds the deciphered version of the character by calling the **`findLetterIndex`** function, and writes it into **deciphered.txt**.

<u>**Hint:**</u>

English alphabet and cipher key comparison

a	**b**	**c**	**d**	**e**	**f**	**g**	**h**	**i**	**j**	**k**	**l**	**m**	**n**	**o**	**p**	**q**	**r**	**s**	**t**	**u**	**v**	**w**	**x**	**y**	**z**
x	h	j	m	s	p	r	w	c	g	e	b	q	i	t	n	k	f	l	a	u	z	d	v	o	y

When ciphering a text, according to the cipher key, “x” will be written in place of every “a”, “h” will be written in place of every “b” and so on. Therefore the word “baba” will become “xhxh” when ciphered. When deciphering a ciphered text, “a” will be written in place of every “x”, “b” will be written in place of every “h” and so on.

Project Name: LG15_Q4
File Name: Q4.cpp

`key.txt`

```
xhjmsprwcgebqitnkflauzdvoy
```

`ciphered.txt`

```
oslasmxo awso lxd otu ci aws bchfxfo
```

`deciphered.txt`

```
yesterday they saw you in the library
```

ADDITIONAL QUESTIONS

AQ1.

Write a C program that reads pairs of binary numbers (each with a maximum of 10 digits) from a text file and prints the sum of binary number pairs in the console.

Write the following functions;

- **`fillArray:`** takes an integer number and an array as parameters, fills the array with the digits of the array, and returns it.
- **`sumOfBinary:`** takes two integer arrays as parameters, calculates their sum, puts the digits of the result into another array, and returns it.

Hint: Notice that the result may contain a maximum of 10 or 11 digits. Try to write a function to insert the digits of the given binary in an array.

Project Name: LG15_AQ1
File Name: AQ1.cpp

Example Run:

```
1. sum is 0 0 0 0 0 1 0 0 0 0 0
2. sum is 0 0 0 0 0 0 1 1 1 1 0
3. sum is 0 0 0 0 1 0 1 0 1 0 1
4. sum is 0 0 0 0 0 0 1 1 1 1 0
5. sum is 0 0 0 0 1 0 0 0 0 0 0
```

binary.txt

```
11011 101
1111 1111
11101 111000
10111 111
111101 11
```

AQ2.

In the CTIS department, a student takes 6 courses, in his/her first-year first semester, whose credits are 5, 4, 3, 3, 1, and 2 respectively.
Write a C program that reads the student ID and the letter grades of all 7 courses from the **"courses.txt"** for each student; calculates the student's GPA for that semester; writes the student ID and the GPA to an output file named **"gpa.txt"**; and by using the function **maxPos** displays the maximum GPA and the ID of the student that gets the maximum gpa on the screen as in the example run. **HINT**: Keep the course credits in an array.

- Convert the letter grade to grade point equivalent (using a switch statement),
- Multiply each grade point with its credits,
- Find their sum,
- Divide the sum by the sum of the credits of that semester.

Write the following function;

- **`maxPos:`** that finds the position of the maximum element in a one-dimensional number array.

courses.txt	**Letter Grade Point**	**gpa.txt**
1111 A A- C C+ A- A-	A 4.00	1111 3.27
2222 D+ A- B- B C- A	A- 3.70	2222 2.67
3333 F D B- C A A-	B+ 3.30	3333 1.64
4444 A- B A A- B- D	B 3.00	4444 3.43
5555 B- C- D+ D F A	B- 2.70	5555 1.51
6666 D+ C C+ B- B B+	C+ 2.30	6666 2.17
7777 B- A- C+ C+ B+ A-	C 2.00	7777 2.93
8888 A- B+ C C+ B D+	C- 1.70	8888 2.79
	D+ 1.30	
	D 1.00	
	F 0.00	

Example Run:

```
Maximum GPA is 3.43 and the student id is 4444
```

Project Name: LG15_AQ2
File Name: AQ2.cpp