19CS109 C PROGRAMMING FOR PROBLEM SOLVING - II

source :
https://programskills.wordpress.com

Hours Per Week :

L	T	P	C
3	-	2	4

Total Hours :

L	T	P
45	-	30

WA/RA	SSH/HSH	CS	SA	S	BS
5	30	5	20	5	5

PREREQUISITE COURSE: C Programming for Problem Solving - I

COURSE DESCRIPTION AND OBJECTIVES:

This course is aimed to impart knowledge on advanced concepts of C programming language and problem solving through programming. It covers strings, pointers, static and dynamic data structures, and also file manipulations. At the end of this course, students will be able to design, implement, test and debug complex programs using advanced features.

COURSE OUTCOMES:

Upon completion of the course, the student will be able to achieve the following outcomes:

COs	Course Outcomes	POs
1	Design and implement of string manipulation functions.	3
2	Create of data structure using dynamic memory and manipulation.	3
3	Create of text files with different access permissions and manipulations.	2
4	Apply suitable formatting for I/O data.	1
5	Develop C programs that are understandable, debuggable, maintainable and more likely to work correctly in the first attempt.	3

SKILLS:

- ✓ *Analyse of the problem to be solved.*
- ✓ *Select static or dynamic data structures for a given problem and manipulation of data items.*
- ✓ *Apply of various file operations effectively in solving real world problems.*
- ✓ *Develop C programs that are understandable, debuggable, maintainable and more likely to work correctly in the first attempt.*

UNIT - I **L-9**

STRINGS: Character array, Reading string from the standard input device, Displaying strings on the standard output device, Importance of terminating a string, Standard string library functions.

UNIT - II **L-9**

POINTERS: Declaration, Initialization, Multiple indirection, Pointer arithmetic, Relationship between arrays and pointers, Scaling up - array of arrays, array of pointers, pointer to a pointer and pointer to an array; Dynamic memory allocation functions.

UNIT - III **L-9**

STRUCTURES: Defining a structure, Declaring structure variable, Operations on structures, Pointers to structure - declaring pointer to a structure, accessing structure members using pointer; Array of structures, Nested structures, Passing structures to functions - passing each member of a structure as a separate argument, passing structure variable by value, passing structure variable by reference/address; Typedef and structures.

UNIT - IV **L-9**

UNIONS: Defining a union - declaring union variable, operations on union; Pointers to union - declaring pointer to a union, accessing union members using pointer; Array of union, Nested union, Typedef and union, Enumerations, Bit-fields.

UNIT - V **L-9**

FILES: Introduction to files, Streams, I/O using streams – opening a stream, closing stream; Character input, Character output, File position indicator, End of file and errors, Line input and line output, Formatted I/O, Block input and output, File type, Files and command line arguments.

LABORATORY EXPERIMENTS

LIST OF EXPERIMENTS **TOTAL HOURS:30**

Experiment 1:

(a) Write a C program to convert the given text into uppercase text.

Hint: Read a line of text character – by – character and store the characters in a char-type array. Read input characters until end-of-line (EOL) character has been read.

If the character is uppercase ignore it, otherwise convert it into uppercase using the library function toupper().

Example:

Hello Vignan

HELLO VIGNAN

(b) A C Program contains the following array declaration char text[80]; Suppose the following string has been assigned to text as “Programming with C is a creative and challenging activity for engineering graduates”.

Notify the significance of execution of the following command lines in printf():

a) printf(“%s”,text);

b) printf(“%18s”text);

c) printf(“%.18s”,text);

d) printf(“%18.7s”,text);

e) printf(“%-18.7s”,text);

Experiment 2:

(a) Write a C program to read string using gets() function and print the contents of the string.

(b) Write a C program to copy a given string into another string without using standard string handling library function **strcpy()**.

Hint: Read one string as an input and then with the help of loop copy the content of given string into the new string. If the storage space allocated to the new string is less than the given string, entire string will not be copied into the new string.

Example: Consider storage space allocated to new string is 20 and given string length is 30. In this case, your program can only copy 20 characters from given string into the new string.

(c) Write a C program to concatenate two strings without using standard string handling library function **strcat()**.

Experiment 3:

Write a C program to concatenate the characters of the two given strings alternatively.

Hint: If the length of the two strings is equal then concatenate the two strings alternatively otherwise concatenate the remaining characters of the higher length string at the end. Concatenated string is different from the given two strings.

Example: If “hi” and “vignan” are two strings then the concatenated string is “hviignan”.

Experiment 4:

(a) Write a C program to reverse a string without using standard string handling library function and, do not use another array to store the reversed string.

Hint: If a user enters a string “hello”, then on reversing it will be displayed as “olleh”.

(b) Write a C program to find whether the given two strings are same or not.

Hint: User need to enter two strings *s1 and s2* and check whether the two strings are same or not. For example: s1=hello, s2=hello output: YES

Experiment 5:

Write a C program to remove blank spaces in the given string.

Input: Hello world

Output: Helloworld

Hint: Read the input through command line arguments. Removal of spaces should be performed on the given string itself.

Experiment 6:

Write a C program for the following:

Given a string S consisting of uppercase and lowercase letters, change the case of each alphabet in this string. That is, all the uppercase letters should be converted to lowercase and all the lowercase letters should be converted to uppercase.

Input: Vignan University

Output: vIGNAN uNIVERSITY

Experiment 7:

Lilly joined a social networking site to stay in touch with her friends. The signup page required the input as name and password. However, the password must be strong. The website considers a password to be strong if it satisfies the following criteria:

a. Its length is at least 6.

b. It contains at least one digit.

c. It contains at least one lowercase/ uppercase English character.

d. It contains at least one special character. The special characters are: !@#$%^&*()-+

She typed a random string of length *n* in the password field but wasn't sure if it was strong. Given the string she typed, can you find the minimum number of characters she must add to make her password strong?

Note: Here's the set of types of characters in a form you can paste in your solution:

Digits = "0123456789"

Lower_case = "abcdefghijklmnopqrstuvwxyz"

upper_case = "ABCDEFGHIJKLMNOPQRSTUVWXYZ"

special_characters = "!@#$%^&*()-+"

Input Format

- The first line contains an integer *n* denoting the length of the string.
- The second line contains a string consisting of *n* characters, the password typed by Louise.
- Each character is either a lowercase/uppercase English alphabet, a digit, or a special character.

Sample Input 0

3

Ab1

Sample Output 0

Password is not strong:

Length should be more than 6

Sample Input 1

12

#HelloVignan

Sample Output 1

Password is not strong:

Password should consists atleast one numeral

Experiment 8:

Write a C program to insert a given character at the beginning and end of the given string.

Hint: If the input string is “C program” and the given character to insert is “g”.

Input: “C program”

Output: “gC programg”

Experiment 9:

Write a C Program to find the frequency of occurrence, of a given character in the given string.

Hint: Read a string and a character to be checked. Then count how many times that the given character has been repeated in the given string.

Example: The given string is: Chinthu, find the frequency of the occurrence of character ‘h’ in the given string. The frequency of occurrence ‘h’ in the given string is 2.

Experiment 10:

Write a C program to insert a character in a specified location of the given string.

Hint: Traverse the string upto the specified location, move the remaining characters back by one position and insert the given character at the specified location.

Example: If given string is ‘Vignan, insert a character at 1st location and the given character is ‘c’. Then the expected output is ‘cVignan’.

Experiment 11:

(a) Write a C program to access the elements of the array using pointers.

Hint: Declare a pointer variable and assign the base address of the array to it and print the values of an array using pointer variable.

(b) Write a C program to count the number of vowels and consonants in a string using pointers.

Hint: Use pointers to read the content of string.

(c) Declare a character array to hold the input string and declare a character pointer. Assign the character array base address to the pointer and then display the every element of the character array.

Hint: Increment the pointer in loop.

Experiment 12:

Create a jagged array (adjacency list representation of a graph) with no of rows and no of columns in each row as specified by the user

Hint: Use Dynamic memory allocation (malloc() or calloc())

Input:

Enter no of rows: 3

Enter no of columns Row in 1: 3

Enter no of columns Row in 2: 5

Enter no of columns Row in 3: 2

Enter the elements row wise:

8 6 5

8 4 6 9 7

9 2

Output:

8 6 5

8 4 6 9 7

9 2

Experiment 13:

Write a C program for the following:

Ram wanted to increase his typing speed to participate in programming contests. His friend suggested that type the sentence "The quick brown fox jumps over the lazy dog" repeatedly. This sentence is known as a pangram because it contains every letter of the alphabet.

After typing the sentence several times, Ram became bored with it so he started to look for other pangrams.

For this task, read a sentence from the user and store it in a character array '*s*'

Hint: Allocate memory for the string using dynamic memory allocation and determine whether the given string is a pangram or not. Ignore upper or lower cases.

Experiment 14:

Write a C program to implement the following:

Define a structure named 'Complex' consisting of two floating point members called "real and imaginary". Let c1 and c2 are two Complex structure variables; compute the sum of two variables.

Experiment 15:

Write a C program for the following:

Customer billing system is a structure, having customers_name, street_address, city, state, account_number, payment_status(paid/ not_paid), payment_date(current date/ due_date), and amount as members. In this example, payment_date is also structure includes month, day and year as members. So, every customer record can be considered as an array of structures. Display the payment status of each customer.

Hint: Use nested structure concept.

Experiment 16:

Write a C program to read the contents character by character from the given text file and display the contents on the standard output device.

Hint: The program makes use of the library functions getc() and putchar() to read and display the data.

Experiment 17:

Write a C program to find whether the given word is present in the given file or not.

Example: The content of the file is "Computer programming. Computer can do computations".

Input: Computer

Output: 'Computer' is found at two locations

Experiment 18:

(a) Write a C program to count the number of characters, number of lines and number of words in a given file.

Hint: Open a text file in read mode and count number of characters, number of lines and number of words in that file.

(b) Write a C program store the data in a text file.

Hint: Open a text file in write mode and read name, roll no and marks of *n* number of students from user and store the above details in the text file.

Experiment 19:

Write a C program to merge two files.

Hint: To merge two files in C programming, first open two files and start copying the content of the first file into the third file(target file) after this start appending the content of the second file into the third file (target file).

TEXT BOOKS:

1. Ajay Mittal, "Programming in C - A practical Approach", 1st edition, Pearson Education Publishers, India, 2010.
2. Reema Thareja, "Introduction to C Programming", 2nd edition, Oxford University Press India, 2015.

REFERENCE BOOKS:

1. Behrouz A. Forouzan, Richard F.Gilberg, "Programming for Problem Solving". 1st edition, Cengage Publishers, 2019.
2. Byron S Gottfried, "Programming with C", 4th edition, Tata McGraw-Hill Publishers, 2018.
3. Herbert Schildt, "C: The Complete Reference", 4th edition, Tata McGraw-Hill, 2017.