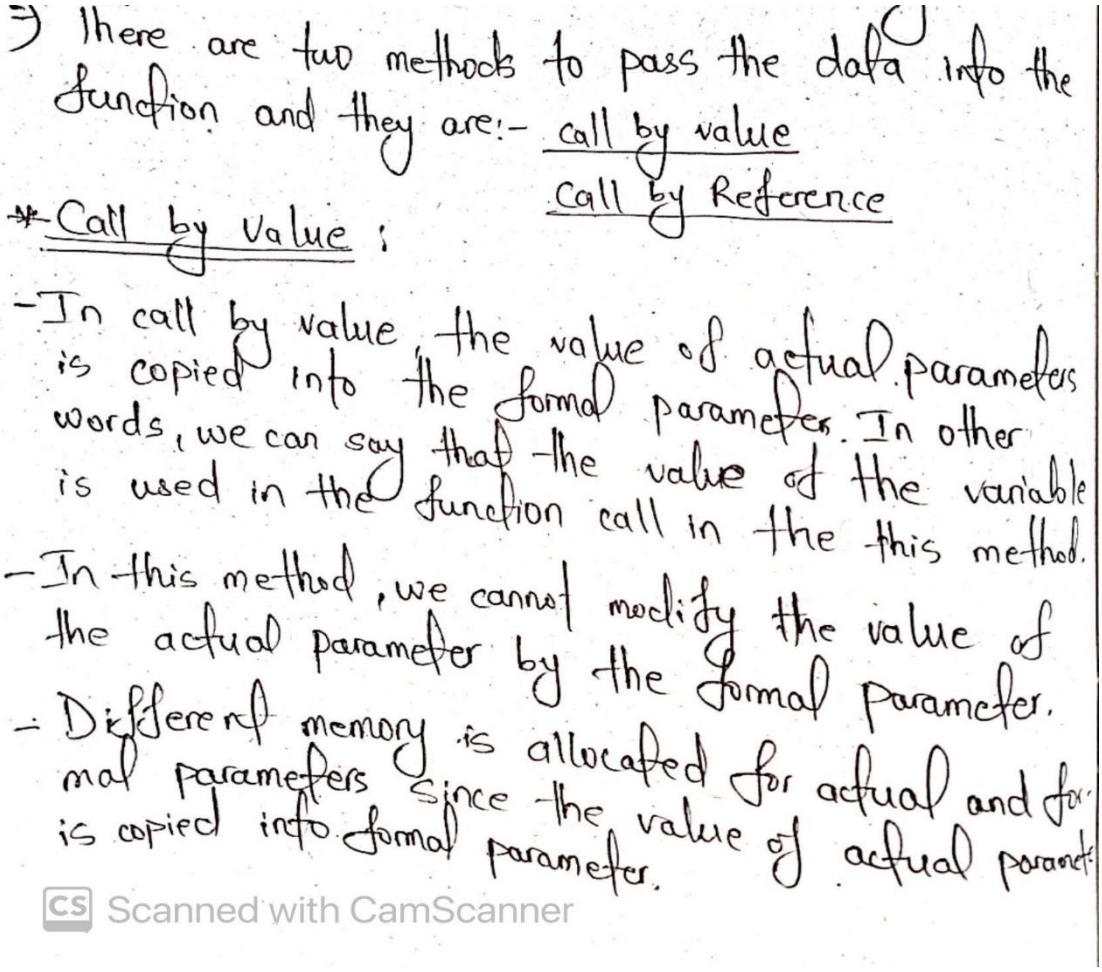
CSE Assignment – 4

Name : Prashant Dhimal

Registration number : AP22110011492

Section : 'W'

1. **Explain about call by value and call by reference with suitable examples.**



There are two methods to pass the data into the function and they are:- call by value
call by Reference

* Call by Value :

- In call by value, the value of actual parameters is copied into the formal parameter. In other words, we can say that the value of the variable is used in the function call in the this method.
- In this method, we cannot modify the value of the actual parameter by the formal parameter.
- Different memory is allocated for actual and formal parameters since the value of actual parameter is copied into formal parameter.

Call by value Example – Swapping two variables.

```
#include <stdio.h>
void swap (int, int);
int main ()
{ int a = 10;
  int b = 20;
printf ("In Before swapping a=%d, b=%d", a, b);
swap (a, b);
printf ("\n After swapping in main a=%d, b=%d", a, b);
}
void swap ( int, int)
{
  int temp;
  temp = a;
  a = b;
  b = temp;
Printf ("\n After swapping in function a=%d, b=%d",
        a, b);
}
```

* Call by Reference

- The address of the variable is passed into the function call as the actual parameter.
- The value of the actual parameters can be modified by changing the formal parameters since the address of the actual parameters is passed.

Call by Reference Example - Swaping

```
#include<stdio.h>
void swapx(int*, int*);
int main()
{
  int a=10, b=20;
  swapx(&a, &b);
  printf("a=%d, b=%d", a, b);
  return 0;
}
void swapx(int *x, int *y)
{
  int t;
  t= *x;
 *x = *y;
 *y = t;
 printf("\n x=%d, y=%d", *x, *y);
}
```

2. Write a C program for Multiplication of two matrixes.

Assignment - 3 > C multiplication_matrix.c > main()

```c
/* C program for multiplication of two matrices in C.*/

#include<stdio.h>
int main()
{
  int r, c, i, j,k,x,y;
  printf("Enter the number of rows for 1st matrix: ");
  scanf("%d", &r);
  printf("Enter the number of columns for st matrix: ");
  scanf("%d", &c);
  printf("Enter the number of rows for 2nd matrix: ");
  scanf("%d", &x);
  printf("Enter the number of columns for 2nd matrix: ");
  scanf("%d", &y);
  int a[r][c], b[r][c], mul[r][c];
  printf("\nEnter elements of 1st matrix:\n");
  for (i = 0; i < r; i++)
{
    for (j = 0; j < c; j++)
    {
      printf("Enter element a%d%d: ", i + 1, j + 1);
      scanf("%d", &a[i][j]);
    }
}

  printf("Enter elements of 2nd matrix:\n");
  for (i = 0; i < x; i++)
{
    for (j = 0; j < y; j++)
    {
      printf("Enter element b%d%d: ", i + 1, j + 1);
      scanf("%d", &b[i][j]);
    }
}
  if (c==x && r==y)
  {
  for (i = 0; i < r; i++)
    {
    for (j = 0; j < c; j++)
      {

        mul[i][j] = 0;
        for (k =0;k<c;k++)
        {
        mul[i][j] += a[i][k] * b[k][j];
        }
      }
    }
  }

  printf("\nMultiplication of two matrices: \n");
  for (i = 0; i < r; i++)
{
    for (j = 0; j < c; j++)
    {
      printf("%d\t", mul[i][j]);
      /*if (j == c - 1)
      {
        printf("\n\n");
      }*/
    }
    printf("\n");
}

  return 0;
}
```

3. Write a C program to implement Fibonacci series using recursion.

Assignment-4 > fibo_recursion.c > main(void)

```c
/*Write a C programme to implement Fibonacci series
using recursion.*/

#include<stdio.h>
int fibonacci(int num);
int main(void)
{
int terms;
printf("Enter terms: ");
scanf("%d", &terms);
for(int n = 0; n < terms; n++)
{
printf("%d ", fibonacci(n));
}
return 0;
}
int fibonacci(int num)
{

if(num == 0 || num == 1)
{
return num;
}
else
{
return fibonacci(num-1) + fibonacci(num-2);
}
}
```

PROBLEMS OUTPUT TERMINAL DEBUG CONSOLE

```
PS P:\vs code> cd "p:\vs code\Assignment-4\" ; if ($?) { gcc fibo_r
Enter terms: 8
0 1 1 2 3 5 8 13
PS P:\vs code\Assignment-4>
```

⇨

4. Explain about String handling functions?

4) Describe and explain different built-in string handling functions with example.

⇒ String in C programming is a sequence of character terminated with a null character '\0'.

(i) strlen() ⇒ It return's the string length.

E.g.
```
#include <stdio.h>
#include <string.h>
int main()
{
 char a[100] = "Prashant";
 printf("\n length of string is %d", strlen(a));
}
```

(ii) strcmp() ⇒ It compares two string and returns 0 if the string are same.

E.g.
```
#include<stdio.h>
#include <string.h>
int main()
{
 char a[100] = "abcd", b[100] = "Abcd", result;
 result = strcmp(a,b);
 printf("\n strcmp(a,b) = %d", result);
}
```

(iii) strcat() ⇒ It concatenates two strings and return the concatenated string.

E.g.
```
#include<stdio.h>
#include <string.h>
int main()
{
char a[100] = "My name is ", b[] = "Prashant";
Printf(" Joining two string = %s ", strcat(a,b));
}
```

(iv) strcpy() ⇒ It copies one string to another

E.g.
```
#include<stdio.h>
#include <string.h>
int main()
{
  char a[100]="Prashant", b[100];
  strcpy(b,a);
  Puts(b);
  return 0;
}
```

(v) strrev() ⇒ It reverse the given string

E.g.
```
#include<stdio.h>
#include <string.h>
int main()
{
  char a[100];
  printf("Enter any character = ");
  scanf("%s", a);
  printf("After reverse = %s", strrev(a));
  return 0;
}
```

(vi) strlwr() : It converts string to lower case.

E.g.
```
#include<stdio.h>
#include <string.h>
int main()
```

```
{
  char a[100] = "HELLO";
  printf ("lowercase = %s", strlwr(a));
  return 0;
}
```

(vii) strupr() => It converts string into uppercase.

Eg.

```
#include <stdio.h>
#include <string.h>
int main()
{
  char a[100] = "prashant";
  printf("Uppercase = %s", strupr(a));
  return 0;
}
```

5. Write a C program to sort the given set of strings.

Assignment-4 > C sort_string.c > main()

```c
/*Write a C programmes to sort the given set of strings.*/

#include<stdio.h>
#include<string.h>
int main()
{
    int r,i,j;
    printf("Enter the number of character you want to arrange in Ascending Order : ");
    scanf("%d", &r);

    char name[r][100],temp[100];
    printf("Enter the character :\n ");
    for (i=0;i<r;i++)
    {
        scanf("%s", name[i]);
    }
    for (i=0;i<r;i++)
    {
        for (j=i+1;j<r;j++)
        {
        if (strcmp(name[i],name[j])>0)
          {
            strcpy(temp,name[i]);
            strcpy(name[i],name[j]);
            strcpy(name[j],temp);
          }
        }
    }
    printf("\nIn ascending Order : \n");
    for (i=0;i<r;i++)
    {
        printf("%s\n",name[i]);
    }
}
```

6. What do u mean by a function? Give the structure of user defined function and explain about the arguments and return values.

(6) What is function? Give the structures of functions and explain about the arguments and return values

⇒ Structure of User-defined function

```
#include <stdio.h>
data_type function_name (data_type variable)
int main()
{
    // function call
    ---------
    statements
  return 0;
}
datatype function_name (datatype variable) // function definition
{
  data_type
  ---------
  statement
  return ---;   // return statement
}
```

⇒ Function arguments and Return value:

~~a) function with no arguments and no return value~~

⇒ Function Arguments

⇒ Arguments in C are variables that are used to pass certain values. These values are generally the source of function that require the arguments during the process execution.

7. **Write a program to read, calculate average and print student marks using array of structures.**

Assignment-4 > C arry_of_structures.c > main()

```c
#include<stdio.h>
struct student
{
    int st_no;
    int st_name[10];
    int m1,m2,m3,avg;
};
void main()
{
    struct student s[60];
    int n,i;
    printf("Enter no of students : ");
    scanf("%d",&n);
    for(int i=1;i<=n;i++)
    {
        printf("\nEnter student number : ");
        scanf("%d",&s[i].st_no);
        printf("\nEnter student name : ");
        scanf("%s",&s[i].st_name);
        printf("\nEnter m1, m2 and m3 : ");
        scanf("%d%d%d",&s[i].m1,&s[i].m2,&s[i].m3);
    }
    printf("\n--------------Students Information--------------\n");
    for (int i=1;i<=n;i++)
    {
        printf("\nStudent no : %d \nStudent Name : %s \nStudent Marks :\nm1 : %d \nm2 : %d \nm3 : %d", s[i].st_no, s[i].st_name, s[i].m1, s[i].m2, s[i].m3);
    }
    for(i=1;i<=n;i++)
    {
        s[i].avg = (s[i].m1+s[i].m2+s[i].m3)/3;
        printf("\nStudent no : %d \nStudent Name : %s \nStudent Average Marks : %d ", s[i].st_no, s[i].st_name, s[i].avg);
    }

}
```

8. Differentiate between self-referential structure and nested structure with example.

(8) Differ^n betw^n self-rederential structure and nested structure, with Example.

Self-Referential	Nested Structure
(i) At least one ~~pointer~~ structure member is a pointer to the structure of its own type.	(i) One structure is member of other structure.
(ii) Useful to create data structure like linked lists, stack, etc.	(ii) Used to access other local function, classes or structure
(iii) Syntax: Example struct node { int data; struct node * next; };	(iii) Syntax: Example: struct date { int day, month, year; }; struct student { int roll; struct date. birth_day; };

9. Explain three dynamic memory allocation functions with suitable

10. **Explain about storage classes.**

(9) Explain three dynamic memory allocation functions with suitable e.g.

=) 3 dynamic memory allocation functions are:-

(i) malloc() : "Malloc" or "memory allocation function" is the method in C used to dynamically allocate a single large block of memory with specified size.

for e.g.

```
ptr = (int *) malloc (n * sizeof(int));
```

(ii) calloc() : "calloc" or "contiguous allocation" method in C is used to dynamically allocate the specified number of blocks of memory of the specified type.

for e.g.:

```
ptr = (float *) calloc (n, sizeof(float));
```

(iii) free() : "free" method in C is used to dynamically de-allocate the memory.

for e.g.

```
free (ptr);
```

(10) Explain Storage class:

a) Storage class in C are used to determine the lifetime, visibility, memory location. and initial value of a variable.

There are 4 types of storage class:

- Automatic: It allocates memory automatically at runtime.
  Automatic variables are initialized to garbage by default.
- Static: The variables defined as static specifier can hold their value betwn the multiple function call.
- Register: Register allocates the memory into the CPU registers depending upon the size of the memory remaining in the CPU.
- External: It is used to tell the compiler that the variable defined as extern is declared with an external linkage elsewhere in the program.

11. Develop a program to create a library catalogue with the following members: access number, authors name, title of the book, year of publication and book price using structures.

Assignment-4 > library_catalouge_structure.c > main()

```c
/*Develop a programme to create a library catalogue
with the following members: access number, authors
name, title of the book, year of publication and book
price using structures.*/

#include<stdio.h>
struct library
{
    int ac_no, year_pub, b_price;
    char name[100], b_title[100];
}s[100];
int main()
{
    int n,i;
    printf("Enter number of members : ");
    scanf("%d",&n);
    for(i=0;i<n;i++)
    {
        printf("Enter the access number for %d member : ",i+1);
        scanf("%d",&s[i].ac_no);
        printf("Enter the authors name for %d member : ",i+1);
        scanf("%s",&s[i].name);
        printf("Enter the title of book for %d member : ",i+1);
        scanf("%s",&s[i].b_title);
        printf("Enter year of publication of book for %d member : ",i+1);
        scanf("%d",&s[i].year_pub);
        printf("Enter price of book for %d member : ",i+1);
        scanf("%d",&s[i].b_price);
    }
    printf("--------------------Library Catalouge--------------------");
    for (i=0;i<n;i++)
    {
        printf("\nAccess Number : %d\nAuthors Name : %s\nTitle of Book : %s\nYear of Publication : %d\nBook Price : %d", s[i].ac_no, s[i].name, s[i].b_title, s[i].year_pub, s[i].b_price);
    }
}
```

12.Explain about command line arguments with an example.

(12) Explain about command line arguments with an example.

=) Command-line arguments are simple parameters that are given on the system's command line and the values of these arguments are passed on to your program during program execution. When a program starts execution without user interaction, command-line arguments are used to pass value or files to it.

=) Command line arguments are passed to the main function as argc and argv.

=) Command line arguments are used to control the program from outside.

=) argv[argc] is a Null pointer.

13. What is a Pointer? Explain pointer arithmetic operations with suitable examples.

(13) What is a pointer? Explain pointer arithmetic operations with suitable e.g.

=) A pointer is a variable that stores the memory address variable as its value.
A pointer variable points to a data type (like int) of the same type, and is created with the * operator.

=) Pointer arithmetic operations are:-

(c) Subtraction:- When a pointer is subtracted with a value, the value is first multiplied by the size of the datatype and then subtracted from the pointer.

e.g:-
```
int n=1;
int *ptr, *ptr1;
ptr=&n; = ptr1;
//subtract
*ptr1 = ptr1-1;
```

(a) Increment / Decrement of a pointer

i) **Increment:** It is a condition that also comes under addition. When a pointer is incremented, it actually increments by no. equal to size of datatype.

E.g. In integer ~~pointer~~ pointer stores address 1000 is incremented, then it will increment by 4 (size of int) and new address will be 1004.

ii) **Decrement:** As above, but is opposite to it

E.g when it is decremented its new address will be 996. (int type)

It will substract the address by the value of its datatype.

b) Addition:

i) When a pointer is added with a value, the value is first multiplied by the size of data type and then added to pointer.

E.g.
```
int n=4;
int *ptr1, *ptr2;
ptr1 = &n;
ptr2 = &n;
// sum
ptr2 = ptr2 + 3;
```

14. What is a file? Explain different modes of opening a file.

(14) What is file ? Explain different modes of opening a file.

=) "A file is a collection of data that is stored permanently on disk". Or, "A file is a stream of bytes of arbitary length, which is used to hold data".

=) Different modes of opening a file are:-

- (read-only) r — open an existing file for reading only.
- (write-only) w — opean a new file for writing only.
- (append-only) a — Open an existing file for appending i.e. to add new information at the end of file.
- r+ (read and write) - Open existing file for update i.e for both reading and writing
- w+ (write and read) - Open new file for both writing and reading
- a+ (append & read) - Open an existing file for both append and reading.

15.Write a program to demonstrate read and write operations on a file.

file_handling.c

Assignment-4 > file_handling.c > main()

```c
/*Write a programme to demonstrate read and write
operations on a file.*/

#include<stdio.h>
#include<stdlib.h>
int main()
{
    char name[100];
    FILE *fptr;

    fptr = fopen("P:\\Program.txt","w");
    if (fptr == NULL)
    {
        printf("Error!");
        exit(1);
    }
    else
    {
        printf("Enter any characters : ");
        scanf("%s",&name);
    }

    fprintf(fptr, "%s", name);
    fclose(fptr);

    if ((fptr = fopen("P:\\Program.txt","r")) == NULL){
       printf("Error! opening file");
       // Program exits if the file pointer returns NULL.
       exit(1);
   }
   fscanf(fptr,"%s", &name);
   printf("Character in Program.txt is = %s", name);
   fclose(fptr);
}

```

16. Explain about fscanf(),fgets(),fprintf() and fwrite() functions with suitable examples.

AND

18. **Explain different file handling functions with syntaxes and suitable examples.**

⇨ Different file handling functions are listed below:

a. fopen() opens new or existing file
b. fprintf() write data into the file
c. fscanf() reads data from the file
d. fputc() writes a character into the file
e. fgetc() reads a character from file
f. fclose() closes the file
g. fseek() sets the file pointer to given position
h. fputw() writes an integer to file
i. fgetw() reads an integer from file
j. ftell() returns current position
k. rewind() sets the file pointer to the beginning of the file

⇨ Examples of all file handling functions are as follows:

Assignment-4 > C file_handling_example.c > ...

```c
//fopen and fprintf and fclose

#include <stdio.h>
main(){
   FILE *fp;
   fp = fopen("file.txt", "w");//opening file
   fprintf(fp, "Hello file by fprintf...\n");//writing data into file
   fclose(fp);//closing file
}


//fscanf
#include <stdio.h>
main(){
   FILE *fp;
   char buff[255];//creating char array to store data of file
   fp = fopen("file.txt", "r");
   while(fscanf(fp, "%s", buff)!=EOF){
   printf("%s ", buff );
   }
   fclose(fp);
}


//fputc() and fgetc()
#include <stdio.h>
main(){
   FILE *fp;
   fp = fopen("file1.txt", "w");//opening file
   fputc('a',fp);//writing single character into file
   fclose(fp);//closing file
}

#include<stdio.h>
void main(){
FILE *fp;
char c;
clrscr();
fp=fopen("myfile.txt","r");

while((c=fgetc(fp))!=EOF){
printf("%c",c);
}
fclose(fp);

}
```

```c
//fputs() and fgets()
#include<stdio.h>
void main(){
FILE *fp;
clrscr();

fp=fopen("myfile2.txt","w");
fputs("hello c programming",fp);

fclose(fp);

}

#include<stdio.h>   //fgets()
void main(){
FILE *fp;
char text[300];
clrscr();

fp=fopen("myfile2.txt","r");
printf("%s",fgets(text,200,fp));

fclose(fp);

}

//fseek()
#include <stdio.h>
void main(){
   FILE *fp;

   fp = fopen("myfile.txt","w+");
   fputs("This is javatpoint", fp);

   fseek( fp, 7, SEEK_SET );
   fputs("sonoo jaiswal", fp);
   fclose(fp);
}
```

17.Write a program to copy one file contents to another.

⇨

Assignment-4 > C copy_1_file_to_another.c > main()

```c
/*Write a programme to copy one file contents to
another.*/

#include <stdio.h>
#include <stdlib.h>

int main()
{
    FILE *fptr1, *fptr2;
    char filename[100], c;

    printf("Enter the filename to open for reading \n");
    scanf("%s", filename);

    // Open one file for reading
    fptr1 = fopen("P:\\Program.txt", "r");
    if (fptr1 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }

    printf("Enter the filename to open for writing \n");
    scanf("%s", filename);

    // Open another file for writing
    fptr2 = fopen("P:\\Program1.txt", "w");
    if (fptr2 == NULL)
    {
        printf("Cannot open file %s \n", filename);
        exit(0);
    }

    // Read contents from file
    c = fgetc(fptr1);
    while (c != EOF)
    {
        fputc(c, fptr2);
        c = fgetc(fptr1);
    }

    printf("\nContents copied to %s", filename);

    fclose(fptr1);
    fclose(fptr2);
    return 0;
}
```