LAB TASKS

**TASK 01:**

**Write a program to open a text file and count the total number of characters, words, and lines in it. Display these counts to the user.**

**Source code:**

#include<stdio.h>

int main()

{

FILE \*file = fopen("m4.txt","r");

if(file==NULL)

{

printf("File doesn't exist");

return 0;

}

int characters=0,lines=0,words = 0;

char ch;

while((ch=fgetc(file))!=EOF)

{

characters++;

if(ch=='\n')

lines++;

if(ch==' ' || ch=='\n')

words++;

}

if(characters>0 && ch!='\n')

lines++;

if(characters>0 && ch!='\n')

words++;

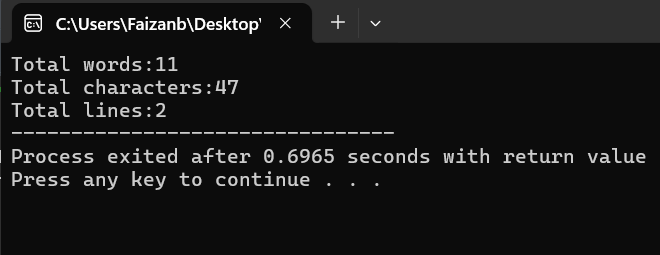
fclose(file);

printf("Total words:%d\n",words);

printf("Total characters:%d\n",characters);

printf("Total lines:%d",lines);

}



**TASK 02:**

**Develop a payroll system to store employee details and their salary structure. Each**

**employee has an ID, name, and a nested salary structure with basic pay, bonuses, and**

**deductions. Write functions to calculate net salary and display employee salary details.**

**Source code:**

#include<stdio.h>

struct Salary{

float basic\_pay;

float bonus;

float deductions;

};

struct Employee{

int id;

char name[20];

struct Salary salary;

};

int calculate\_net\_salary(struct Salary salary)

{

return salary.basic\_pay + salary.bonus + salary.deductions;

}

void display\_details(struct Employee emp)

{

float netsalary = calculate\_net\_salary(emp.salary);

printf("Employee id:%d\n",emp.id);

printf("Employee name:%s\n",emp.name);

printf("Baasic pay:%.2f\n",emp.salary.basic\_pay);

printf("Bonus:%.2f\n",emp.salary.bonus);

printf("Deductions:%.2f\n",emp.salary.deductions);

printf("Net Salary:%.2f",netsalary);

}

int main()

{

struct Employee emp;

printf("Enter employee id:");

scanf("%d",&emp.id);

printf("Enter employee name:");

scanf(" %[^\n]",emp.name);

printf("Enter basic pay:");

scanf("%f",&emp.salary.basic\_pay);

printf("Enter bonus:");

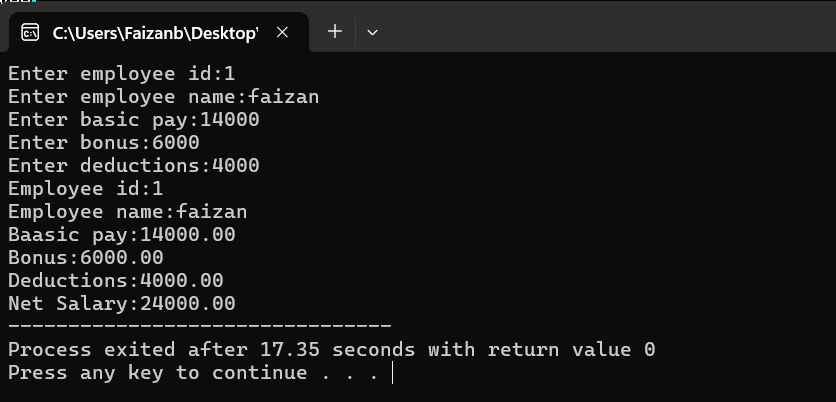
scanf("%f",&emp.salary.bonus);

printf("Enter deductions:");

scanf("%f",&emp.salary.deductions);

display\_details(emp);

}



**TASK 03:**

**Write a program in C to merge two files and write them to another file.**

**Source code:**

#include<stdio.h>

int main()

{

char ch;

FILE \*file1 = fopen("m1.txt","r");

if(file1==NULL)

{

printf("File doesn't exist");

return 0;

}

FILE \*file2 = fopen("m2.txt","r");

if(file2==NULL)

{

printf("File doesn't exist");

return 0;

}

FILE \*mergefile = fopen("merge.txt","w");

if(mergefile==NULL)

{

printf("File doesn't exist");

fclose(file1);

fclose(file2);

return 0;

}

while ((ch = fgetc(file1)) != EOF) {

fputc(ch, mergefile);

}

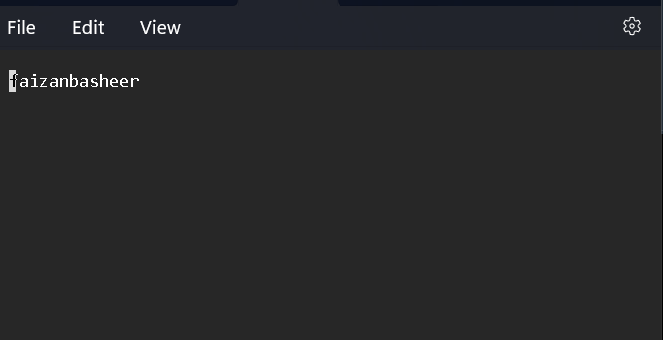
while ((ch = fgetc(file2)) != EOF) {

fputc(ch, mergefile);

}

printf("Files successfully merged");

}



A black rectangular object with a black border

Description automatically generated

A black rectangular object with a white border

Description automatically generated

**TASK 04:**

**Write a program to delete all contents of a file named data.txt. After deleting, confirm the file is empty by attempting to read it back.**

**Source code:**

#include<stdio.h>

int main()

{

FILE \*file = fopen("p1.txt","w");

if(file==NULL)

{

printf("File doesn't exist"); //opening the file in write mode delete it content

return 1;

}

fclose(file);

file = fopen("p1.txt","r");

if(fgetc(file)==EOF)

{

printf("File is empty");

}

else

printf("File is not empty");

}

Before running program

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**TASK 05:**

**Create a structure to manage a sports team and its players. Each team has a name, sport, and an array of players. Each player has a name, age, and position. Implement functions to add players, search by position, and display team details.**

**Source code:**

#include<stdio.h>

#include<string.h>

struct Player{

char name[20];

int age;

char position[10];

};

struct Team{

char name[20];

char sports[10];

struct Player player[10];

int players\_count;

};

void addplayer(struct Team \*team)

{

if(team->players\_count<10)

{

struct Player new\_player;

printf("Enter player name:");

scanf(" %[^\n]",new\_player.name);

printf("Enter player age:");

scanf(" %d",&new\_player.age);

printf("Enter player position:");

scanf(" %[^\n]",new\_player.position);

team->player[team->players\_count] = new\_player;

team->players\_count++;

}

else

{

printf("Team is full.Cannot add more players....");

return ;

}

}

void display\_team\_Details(struct Team \*team)

{

printf("\n");

printf("Team name:%s\n",team->name);

printf("Sport name:%s\n",team->sports);

printf("Players\n");

for(int i = 0 ;i<team->players\_count;i++)

{

printf("%d.Name:%s Age:%d Position:%s\n",i+1,team->player[i].name,team->player[i].age,team->player[i].position);

}

}

void search\_position(struct Team \*team,char \* position)

{

int found = 0;

for (int i = 0; i < team->players\_count; i++) {

if (strcmp(team->player[i].position, position) == 0) {

if (!found) {

printf("Player(s) at position '%s':\n", position);

found = 1;

}

printf("Name: %s, Age: %d, Position: %s\n",

team->player[i].name, team->player[i].age, team->player[i].position);

}

}

if (!found) {

printf("No player found at position '%s'.\n", position);

}

}

int main()

{

struct Team team;

int n;

printf("Enter team name:");

scanf("%[^\n]",team.name);

printf("Enter sports name:");

scanf(" %[^\n]",team.sports);

printf("Enter no of players:");

scanf("%d",&n);

for( int i = 0; i < n; i++)

{

printf("Adding player %d\n",i+1);

addplayer(&team);

}

display\_team\_Details(&team);

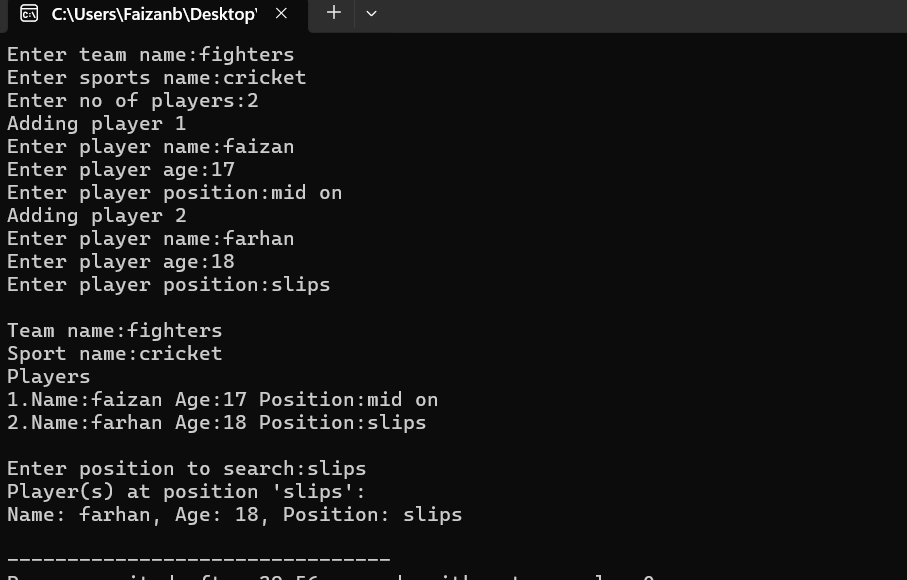
char position[10];

printf("\nEnter position to search:");

scanf(" %[^\n]",position);

search\_position(&team,position);

}



**TASK 06:**

**Create a program that opens a file named log.txt and appends a new line with the current timestamp each time the program runs. Use this file to track each time the program was executed.**

**Source code;**

#include<stdio.h>

#include<time.h>

int main()

{

FILE \*file = fopen("log.txt","a");

if(file==NULL)

{

printf("File doesn't exist..");

return 0;

}

time\_t now = time(NULL);

if(now==-1)

{

printf("Could not get current time");

fclose(file);

return 0;

}

fprintf(file,"%s",ctime(&now));

printf("Time appended successfully");

fclose(file);

}

A screenshot of a computer

Description automatically generated