**Report**

**Title Page**

**Project Title:** Charity Fundraising System  
**Your Name:**  Maria Royce  
**Roll Number:**  51  
**Course Name:** Programming in C  
**Date:** 18 Jul 2024

**Introduction**

**Brief Overview of the Project**

The Charity Fundraising Management System is an application written in C. It is designed to help manage charity fundraising events allowing the users to add events, make donations, and view the progress of all events happening .This program ensures that the data persists between sessions.

**Problem Statement**

This project aims to solve the issues faced in managing charity fund by providing a platform to check the progress, to set a target and to check whether the target has been achieved.

**Objective**

The primary objective of this project is to develop a C program that:

* Allows users to add new events.
* Enables users to donate to specific events.
* Displays the progress of fundraising events
* It also includes the progress of the events.
* Transfer event data to a file for future use.

**System Requirements**

**Hardware Requirements**

* Processor: Intel Core i3 or equivalent
* RAM: 2 GB or more
* Storage: 10 MB of free disk space

**Software Requirements**

* Operating System: Windows, Linux, or macOS
* Compiler: GCC (GNU Compiler Collection) or any standard C compiler
* Text Editor or IDE: Visual Studio Code, Code::Blocks, or any C programming IDE

**Design and Development**

**Description of the Program Logic**

1. **Data Structures:**
   * The program uses a structure (charity) to store information about each fundraising event, including the event name, target amount, and amount raised.
2. **File Operations:**
   * Events are saved to and read from a file named **event.txt** .This file stores the number of events and their details.
3. **Functions:**
   * **read():** Reads event data from ***event.txt*** and initializes the events array.
   * **save():** Writes the current state of events to ***event.txt***.
   * **addEvent():** Adds a new event by updating the events array and saving the changes to the file.
   * **donate():** Allows users to donate to a specific event and updates the event's fund.
   * **viewProgress():** Displays the current progress of all events, including whether the fundraising target has been met.

**Pseudocode**

Brief pseudocode representation of the core logic:

BEGIN

CALL read() // Load events from file

WHILE (TRUE)

DISPLAY menu options

GET user choice

IF choice == 1 THEN

CALL addEvent()

ELSE IF choice == 2 THEN

CALL donate()

ELSE IF choice == 3 THEN

CALL viewProgress()

ELSE IF choice == 4 THEN

EXIT

ELSE

DISPLAY "Invalid choice"

END IF

END WHILE

END

**Testing and Results**

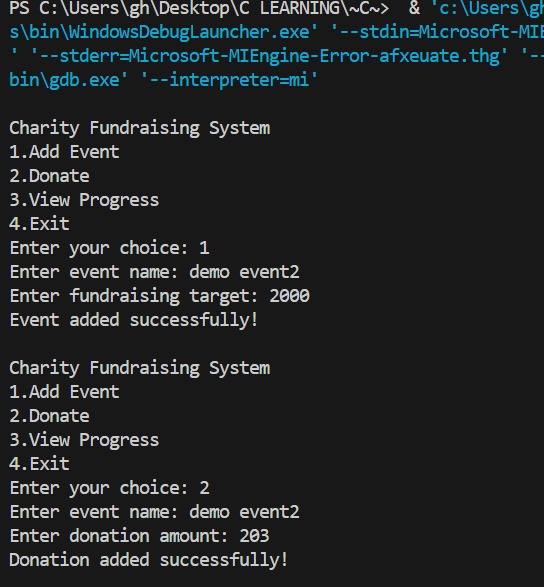
**Test Cases**

* **Add Event:**
  + Input: Event name "demo event2",target amount 2000.
  + Output: Event added successfully, saved to event.txt.
* **Donate:**
* Input: Event name " demo event2", donation amount 203.
* Output: Donation added successfully, updated in event.txt.
* **View Progress:**
  + Input: no input
  + Output: Displays all event progress, including amount raised and percentage of the target achieved.
* **Exception** :
  + Input: Invalid event name.
  + Output: displays warning message.

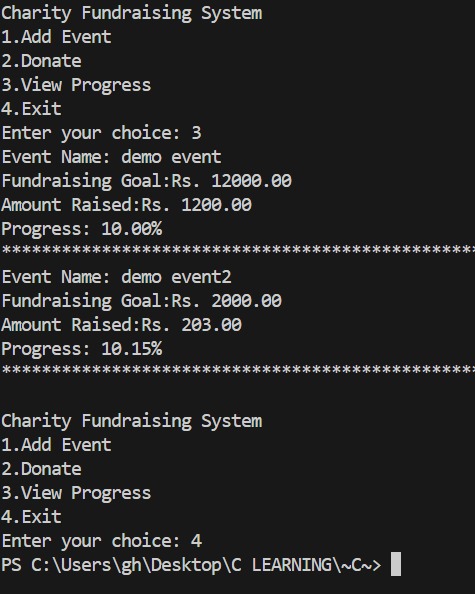
**Output Screenshots or Results**

For each function, the output is as follows:

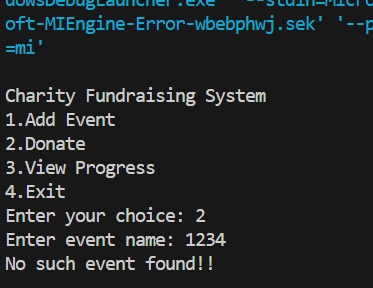
* **Add Event & Donate**



* **View Progress:**



* **Exception**:



**Discussion of Results**

The program meets the objectives by allowing users to manage charity events, make donations, and check progress. Data persists with use of file operations. User input is processed and results are displayed as expected.

**Conclusion**

**Summary of the Project**

The Charity Fundraising Management System provides a solution for managing charity events. It supports adding new events, making donations, and tracking the progress of fundraising efforts. The use of file operations ensures data is preserved between program runs.

**Future Enhancements**

1. Upgrade the system to handle more events.
2. Implement features to provide details about the event, detail of donor and reports.
3. Upgrade to GUI interface for better user experience.

**References**

* File Handling in C Programming, Online Tutorials

**Appendices**

**Code Listing**

#include <stdio.h>

#include <stdlib.h>

#include <string.h>

typedef struct

{

    char name[100];

    double target;

    double fund;

} charity;

charity events[5];

int eventCount=0;

//Functions~~~~~~~

void read()

{

    FILE \*fp=fopen("event.txt","r");

    if (fp==NULL)

    {

        return;

    }

    fscanf(fp,"%d\n",&eventCount);

    for (int i=0;i<eventCount;i++)

    {

        fgets(events[i].name,100,fp);

        events[i].name[strcspn(events[i].name, "\n")] = 0;

        fscanf(fp,"%lf\n%lf\n",&events[i].target,&events[i].fund);

    }

    fclose(fp);

}

void save()

{

    FILE \*fp=fopen("event.txt","w");

    if (fp==NULL)

    {

        printf("Unable to save events.\n");

        return;

    }

    fprintf(fp,"%d\n",eventCount);

    for (int i=0;i<eventCount;i++)

    {

        fprintf(fp,"%s\n",events[i].name);

        fprintf(fp,"%lf\n",events[i].target);

        fprintf(fp,"%lf\n",events[i].fund);

    }

    fclose(fp);

}

void addEvent()

{

    if (eventCount>=5)

    {

        printf("Maximum number of events reached.\n");

        return;

    }

    printf("Enter event name: ");

    scanf(" %[^\n]s",events[eventCount].name);

    printf("Enter fundraising target: ");

    scanf("%lf",&events[eventCount].target);

    events[eventCount].fund=0;

    eventCount++;

    save();

    printf("Event added successfully!\n");

}

void donate()

{

    char name[100];

    double amount;

    printf("Enter event name: ");

    scanf(" %[^\n]s", name);

    int i;

    for (i = 0;i<eventCount;i++)

    {

        if (strcmp(events[i].name,name)==0)

        {

            printf("Enter donation amount: ");

            scanf("%lf",&amount);

            events[i].fund+=amount;

            save();

            printf("Donation added successfully!\n");

            return;

        }

        printf("No such event found!!\n");

    }

}

void viewProgress()

{

      int i;

    for (i=0;i<eventCount;i++)

    {

            printf("Event Name: %s\n", events[i].name);

            printf("Fundraising Goal:Rs. %.2f\n",events[i].target);

            printf("Amount Raised:Rs. %.2f\n",events[i].fund);

            if (events[i].fund>=events[i].target) {

                printf("Goal achieved!\n");

            }

            else

            {

                printf("Progress: %.2f%%\n",(events[i].fund / events[i].target) \* 100);

             }

             printf("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\n");

    }

}

//Main function~~~~~~~~~~~~~~

int main()

{

    read();

    int choice;

    while (1)

     {

        printf("\nCharity Fundraising System\n");

        printf("1.Add Event\n");

        printf("2.Donate\n");

        printf("3.View Progress\n");

        printf("4.Exit\n");

        printf("Enter your choice: ");

        scanf("%d", &choice);

        switch (choice)

        {

            case 1:

                addEvent();

                break;

            case 2:

                donate();

                break;

            case 3:

                viewProgress();

                break;

            case 4:

                //save();

                exit(0);

                break;

            default:

                printf("Invalid choice. Please try again.\n");

        }

    }

    return 0;

}