VISVESVARAYA TECHNOLOGICAL UNIVERSITY
"Jnana Sangama", Belagavi, Karnataka

A Mini project on

"Personal Expense Tracker"

Submitted in partial fulfillment for Mini Project Report
In
" Principles of Programming using C (BPOPS103) "

of
BACHELOR OF ENGINEERING
In
COMPUTER SCIENCE AND ENGINEERING

Submitted by

C Tejesh (A41)

Anraj(A29)
B. Hari kishan(A35)

Varun reddy(A62)

RV INSTITUTE OF TECHNOLOGY AND MANAGEMENT®
(Affiliated to Visvesvaraya Technological University, Belagavi & Approved by AICTE, New Delhi)
Chaitanya Layout, JP Nagar 8th Phase, Kothanur, Bengaluru-560076

ABSTRACT

This mini project, **Personal Expense Tracker**, is a C-based program designed to help users manage their personal finances efficiently. The application features functionalities to add, edit, and delete expenses, categorize them, and store data persistently using file handling. This project demonstrates fundamental programming concepts such as modular design, file handling, and menu-driven interfaces. It provides a user-friendly approach to track expenses and analyze financial data, making it an effective tool for personal finance management.

INTRODUCTION

The **Personal Expense Tracker** mini project is a C programming application aimed at providing users with an intuitive platform to manage their expenses. The project employs file handling to store and retrieve expense data, ensuring that users can access their financial records even after exiting the application.

Features:

1. **Add Expense**: Users can input expenses, specifying the amount, description, and category.
2. **Edit Expense**: Modify existing records by selecting the desired entry.
3. **Delete Expense**: Remove specific expenses or clear all expenses.
4. **Categorize Expenses**: Organize expenses into predefined or user-defined categories.
5. **File-Based Storage**: Persist expense data for future sessions.

Objectives:

- To implement a practical solution for personal finance tracking using C programming.
- To demonstrate the usage of modular programming and file handling techniques.
- To provide an application that is user-friendly and efficient in handling financial data.

This project emphasizes programming best practices such as modularity, error handling, and user feedback, making it a robust application suitable for personal use.

CODE

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

// Define constants for categories
#define MAX_CATEGORIES 100
#define MAX_ENTRIES 100

char categories[MAX_CATEGORIES][20] = {"Food", "Transport", "Entertainment",
"Bills", "Others"};
int categoryCount = 5;

// Define a structure to hold expense data
typedef struct {
    char category[20];
    float amount;
} Expense;

// Function prototypes
void addCategory();
void addExpense(Expense expenses[], int *count, const char *filename);
void displayExpenses(const char *filename);
void displayTotalByCategory(const char *filename);
void saveExpenseToFile(Expense expense, const char *filename);
void removeExpense(const char *filename);
void removeAllExpenses(const char *filename);
void editExpense(const char *filename);

int main() {
    Expense expenses[MAX_ENTRIES];
    int expenseCount = 0;
    int choice;
    const char *filename = "expenses.txt";
```

```
do {
   printf("\n--- Personal Expense Tracker ---\n");
   printf("1. Add Category\n");
   printf("2. Add Expense\n");
   printf("3. Display All Expenses\n");
   printf("4. Display Total by Category\n");
   printf("5. Remove an Expense\n");
   printf("6. Edit an Expense\n");
   printf("7. Remove All Expenses\n");
   printf("8. Exit\n");
   printf("Enter your choice: ");
   scanf("%d", &choice);

   switch (choice) {
      case 1:
         addCategory();
         break;
      case 2:
         addExpense(expenses, &expenseCount, filename);
         break;
      case 3:
         displayExpenses(filename);
         break;
      case 4:
         displayTotalByCategory(filename);
         break;
      case 5:
         removeExpense(filename);
         break;
      case 6:
         editExpense(filename);
         break;
      case 7:
         removeAllExpenses(filename);
```

```
                break;
            case 8:
                printf("Exiting the program. Goodbye!\n");
                break;
            default:
                printf("Invalid choice. Please try again.\n");
        }
    } while (choice != 8);

    return 0;
}

// Function to add a new category
void addCategory() {
    if (categoryCount >= MAX_CATEGORIES) {
        printf("Error: Maximum number of categories reached!\n");
        return;
    }

    printf("Enter the name of the new category: ");
    scanf("%s", categories[categoryCount]);
    categoryCount++;
    printf("Category added successfully!\n");
}

// Function to add a new expense
void addExpense(Expense expenses[], int *count, const char *filename) {
    if (*count >= MAX_ENTRIES) {
        printf("Error: Maximum number of entries reached!\n");
        return;
    }

    printf("\nAvailable Categories:\n");
    for (int i = 0; i < categoryCount; i++) {
        printf("%d. %s\n", i + 1, categories[i]);
    }

    int categoryChoice;
    printf("Enter category number: ");
    scanf("%d", &categoryChoice);

    if (categoryChoice < 1 || categoryChoice > categoryCount) {
        printf("Invalid category. Expense not added.\n");
        return;
    }

    Expense newExpense;
    strcpy(newExpense.category, categories[categoryChoice - 1]);

    printf("Enter amount: ");
    scanf("%f", &newExpense.amount);

    expenses[*count] = newExpense;
    (*count)++;

    saveExpenseToFile(newExpense, filename);

    printf("Expense added successfully!\n");
}

// Function to save an expense to a file
void saveExpenseToFile(Expense expense, const char *filename) {
    FILE *file = fopen(filename, "a");
    if (file == NULL) {
        printf("Error: Could not open file for writing.\n");
        return;
    }

    fprintf(file, "%s %.2f\n", expense.category, expense.amount);
    fclose(file);
```

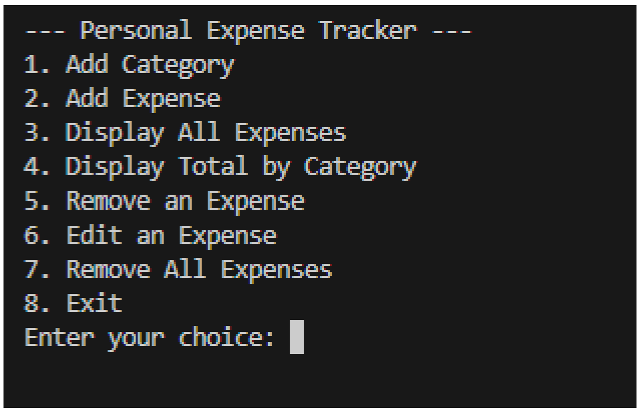
```
}

// Function to display all expenses
void displayExpenses(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("No expenses to display.\n");
        return;
    }

    printf("\nAll Expenses:\n");
    char category[20];
    float amount;
    int count = 1;

    while (fscanf(file, "%s %f", category, &amount) != EOF) {
        printf("%d. Category: %s, Amount: %.2f\n", count++, category, amount);
    }

    fclose(file);
}

// Function to display total expenses by category
void displayTotalByCategory(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("No expenses to display.\n");
        return;
    }

    float totals[MAX_CATEGORIES] = {0};
    char category[20];
    float amount;

    while (fscanf(file, "%s %f", category, &amount) != EOF) {
        for (int i = 0; i < categoryCount; i++) {
            if (strcmp(category, categories[i]) == 0) {
                totals[i] += amount;
                break;
            }
        }
    }

    fclose(file);

    printf("\nTotal Expenses by Category:\n");
    for (int i = 0; i < categoryCount; i++) {
        printf("%s: %.2f\n", categories[i], totals[i]);
    }
}

// Function to remove an expense
void removeExpense(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("No expenses to remove.\n");
        return;
    }

    Expense expenses[MAX_ENTRIES];
    int count = 0;

    while (fscanf(file, "%s %f", expenses[count].category, &expenses[count].amount) !=
EOF) {
        count++;
    }
    fclose(file);

    if (count == 0) {
        printf("No expenses to remove.\n");
        return;
    }

    displayExpenses(filename);
    int index;
    printf("Enter the number of the expense to remove: ");
    printf("(Enter 0 to remove all expenses)\n");
    scanf("%d", &index);

    if (index == 0) {
        removeAllExpenses(filename);
        return;
    }

    if (index < 1 || index > count) {
        printf("Invalid entry.\n");
        return;
    }

    file = fopen(filename, "w");
    if (file == NULL) {
        printf("Error: Could not open file for writing.\n");
        return;
    }

    for (int i = 0; i < count; i++) {
        if (i != index - 1) {
            fprintf(file, "%s %.2f\n", expenses[i].category, expenses[i].amount);
        }
    }
    fclose(file);

    printf("Expense removed successfully!\n");
}
```

```
// Function to remove all expenses
void removeAllExpenses(const char *filename) {
    // Check if the file exists and has content
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("No expenses to remove.\n");
        return;
    }

    // Check if the file is empty
    fseek(file, 0, SEEK_END);
    if (ftell(file) == 0) {
        printf("No expenses to remove.\n");
        fclose(file);
        return;
    }
    fclose(file);

    // Overwrite the file to remove all data
    file = fopen(filename, "w");
    if (file == NULL) {
        printf("Error: Could not open file for writing.\n");
        return;
    }
    fclose(file);

    // Check if the file is now empty and delete it
    file = fopen(filename, "r");
    fseek(file, 0, SEEK_END);
    if (ftell(file) == 0) {
        fclose(file);
        if (remove(filename) == 0) {
            printf("All expenses removed, and the file has been deleted successfully!\n");
        } else {
            printf("All expenses removed, but the file could not be deleted.\n");
        }
    } else {
        fclose(file);
        printf("All expenses removed successfully!\n");
    }
}


// Function to edit an expense
void editExpense(const char *filename) {
    FILE *file = fopen(filename, "r");
    if (file == NULL) {
        printf("No expenses to edit.\n");
        return;
    }

    Expense expenses[MAX_ENTRIES];
    int count = 0;

    while (fscanf(file, "%s %f", expenses[count].category, &expenses[count].amount) !=
EOF) {
        count++;
    }
    fclose(file);

    if (count == 0) {
        printf("No expenses to edit.\n");
        return;
    }

    displayExpenses(filename);
    int index;
    printf("Enter the number of the expense to edit: ");
    scanf("%d", &index);
    if (index < 1 || index > count) {
        printf("Invalid entry.\n");
        return;
    }

    printf("Current Category: %s, Amount: %.2f\n", expenses[index - 1].category,
expenses[index - 1].amount);

    printf("Enter new category: \n");
    for (int i = 0; i < categoryCount; i++) {
        printf("%d. %s\n", i + 1, categories[i]);
    }

    int categoryChoice;
    printf("Enter category number: ");
    scanf("%d", &categoryChoice);

    if (categoryChoice < 1 || categoryChoice > categoryCount) {
        printf("Invalid category. Expense not edited.\n");
        return;
    }

    strcpy(expenses[index - 1].category, categories[categoryChoice - 1]);

    printf("Enter new amount: ");
    scanf("%f", &expenses[index - 1].amount);

    file = fopen(filename, "w");
    if (file == NULL) {
        printf("Error: Could not open file for writing.\n");
        return;
    }

    for (int i = 0; i < count; i++) {
        fprintf(file, "%s %.2f\n", expenses[i].category, expenses[i].amount);
    }
    fclose(file);

    printf("Expense edited successfully!\n");
}
```

OUTPUT

1. **Main Menu :-**

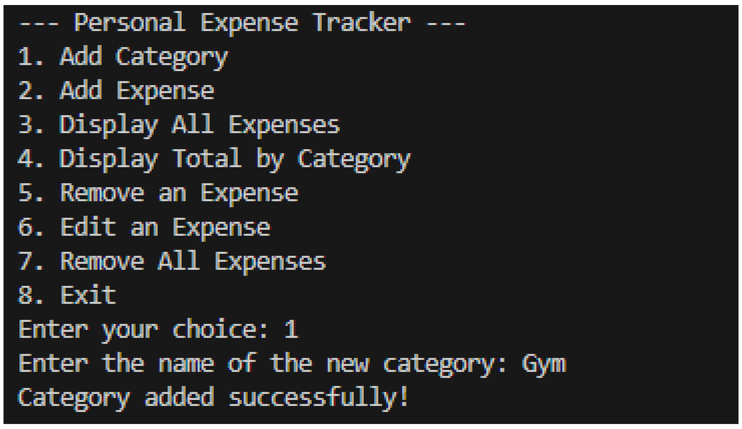
The main menu offers options such as add category, add expense, display all expenses, display total by category, remove an expense, edit an expense, remove all expenses, exit.



```
--- Personal Expense Tracker ---
1. Add Category
2. Add Expense
3. Display All Expenses
4. Display Total by Category
5. Remove an Expense
6. Edit an Expense
7. Remove All Expenses
8. Exit
Enter your choice: 
```

2. **Add Category :-**

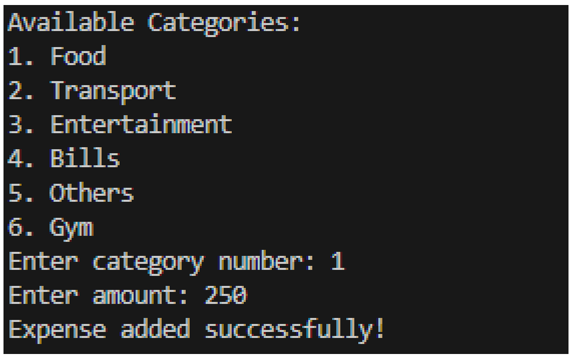
Users can add a new category for their personal expense.



```
--- Personal Expense Tracker ---
1. Add Category
2. Add Expense
3. Display All Expenses
4. Display Total by Category
5. Remove an Expense
6. Edit an Expense
7. Remove All Expenses
8. Exit
Enter your choice: 1
Enter the name of the new category: Gym
Category added successfully!
```

3. **Add Expense:-**

Prompts the user to input the expense details. The expense amount is recorded.

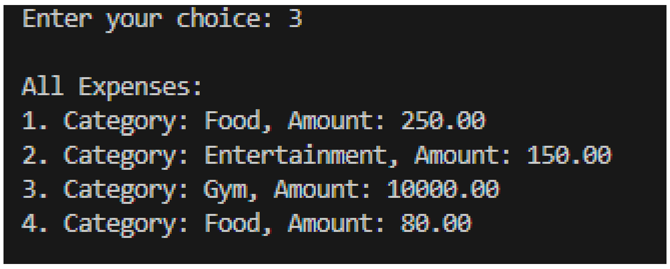


```
Available Categories:
1. Food
2. Transport
3. Entertainment
4. Bills
5. Others
6. Gym
Enter category number: 1
Enter amount: 250
Expense added successfully!
```

4. **Displaying All Expenses:-**

   Reads the expense data from the file and displays it in a tabular format, showing:

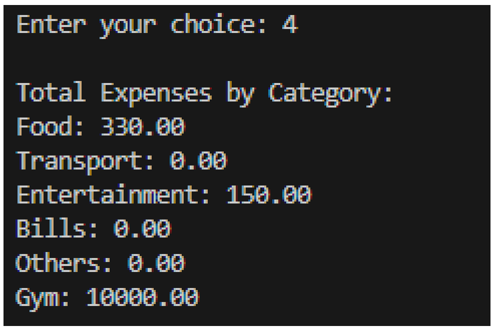
- Expense category.
- Amount spent.



```
Enter your choice: 3

All Expenses:
1. Category: Food, Amount: 250.00
2. Category: Entertainment, Amount: 150.00
3. Category: Gym, Amount: 10000.00
4. Category: Food, Amount: 80.00
```

5. **Displaying Total by Category:-**

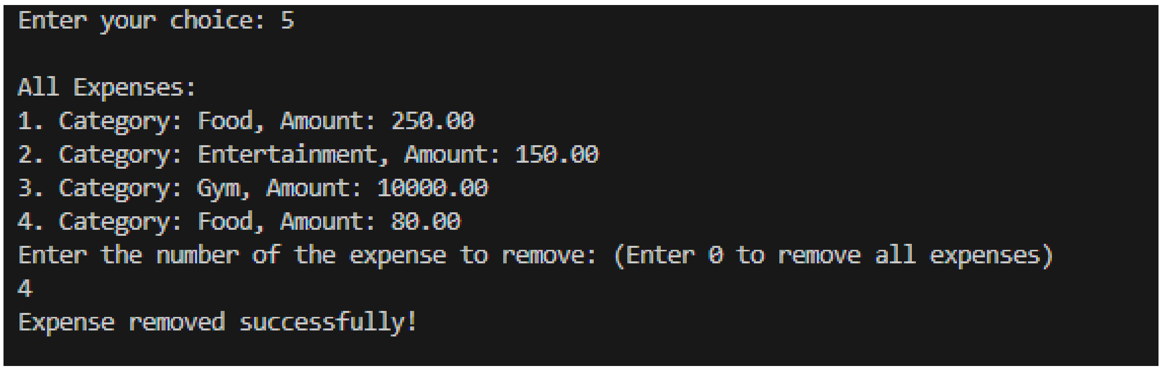
   Reads the expense data from the file and displays it in a tabular format by adding the total amount by category.



```
Enter your choice: 4

Total Expenses by Category:
Food: 330.00
Transport: 0.00
Entertainment: 150.00
Bills: 0.00
Others: 0.00
Gym: 10000.00
```

6. **Remove an Expense:-**

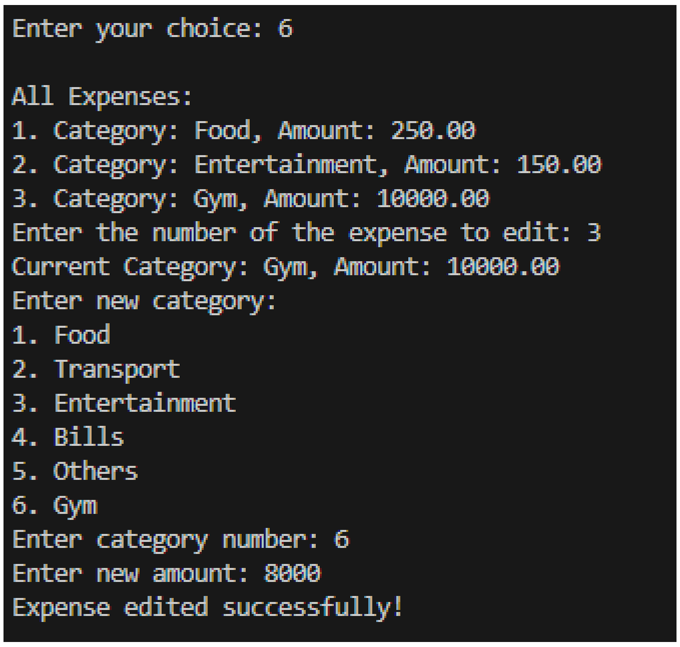
   Users can remove a specific expense by selecting its entry number.



```
Enter your choice: 5

All Expenses:
1. Category: Food, Amount: 250.00
2. Category: Entertainment, Amount: 150.00
3. Category: Gym, Amount: 10000.00
4. Category: Food, Amount: 80.00
Enter the number of the expense to remove: (Enter 0 to remove all expenses)
4
Expense removed successfully!
```

7. **Edit An Expense:-**

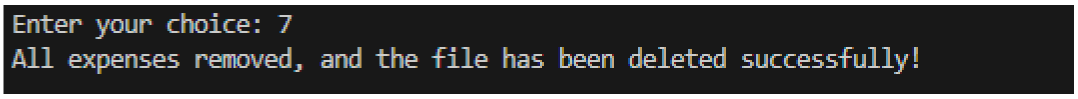
Allows users to modify an existing expense by selecting it from the list of recorded expenses.



```
Enter your choice: 6

All Expenses:
1. Category: Food, Amount: 250.00
2. Category: Entertainment, Amount: 150.00
3. Category: Gym, Amount: 10000.00
Enter the number of the expense to edit: 3
Current Category: Gym, Amount: 10000.00
Enter new category:
1. Food
2. Transport
3. Entertainment
4. Bills
5. Others
6. Gym
Enter category number: 6
Enter new amount: 8000
Expense edited successfully!
```

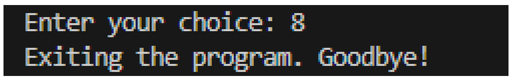
8. **Remove All Expenses:-**

Clears all recorded expenses by overwriting the file and optionally deletes the file if empty.



```
Enter your choice: 7
All expenses removed, and the file has been deleted successfully!
```

9. **Exiting the Program:-**



```
Enter your choice: 8
Exiting the program. Goodbye!
```