EXPENSE TRACKER PROJECT IN C PROGRAMMING LANGUAGE: BLOG

Group Members

IshwariDesai

Sanika Potdar,

Sonal Karche

Building an Expense Tracker in C: A Step-by-Step Guide

Managing finances can be challenging, but with a handy Expense Tracker program, you can keep tabs on your spending, analyse trends, and stick to your budget. In this blog, we'll walk you through a simple yet functional Expense Tracker application written in C. This program allows users to add expenses, view reports, and monitor budgets effectively.

Features of the Expense Tracker Program:

1. **Add Expense:** Users can log new expenses by category, amount, and date.
2. **View Expenses:** Displays a list of all recorded expenses.

3. **Monthly Report:** Generates reports for a specific month and year.
4. **Annual Report:** Summarizes expenses for an entire year.
5. **Budget Setting:** Allows users to set and monitor their budget.

PROGRAM OVERVIEW:

- We start including necessary libraries:

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

1. stdio.h for input and output.
2. stdlib.h for memory allocation (if needed later).
3. string.h for string manipulation.

- **Key Data Structures:**

```c
#define MAX_EXPENSES 100
#define MAX_CATEGORIES 10

typedef struct {
    char category[50];
    float amount;
    char date[11]; // format: YYYY-MM-DD
} Expense;

typedef struct {
    Expense expenses[MAX_EXPENSES];
    int expense_count;
    float budget;
} ExpenseTracker;
```

1. E⍒Ⓟ◯∅Ⓢ◯ structure holds information for each expense (category, amount, and date).

2. ExpenseTracker structure manages a collection of expenses and the user's budget.

- **Main Menu:**

```c
void displayMenu() {
    printf("\nExpense Tracker Menu:\n");
    printf("1. Add Expense\n");
    printf("2. View Expenses\n");
    printf("3. Monthly Report\n");
    printf("4. Annual Report\n");
    printf("5. Set Budget\n");
    printf("0. Exit\n");
}
```

The displayMenu() function presents the user with available actions.

- **Core Functions:**

1. **Adding an Expense:**

Users input the category, amount, and date. The program checks if the total expense exceeds the set budget.

```c
void addExpense(ExpenseTracker *tracker) {

    Expense new_expense;
    printf("Enter category: ");
    scanf("%s", new_expense.category);
    printf("Enter amount: ");
    scanf("%f", &new_expense.amount);
    printf("Enter date (YYYY-MM-DD): ");
    scanf("%s", new_expense.date);

    tracker->expenses[tracker->expense_count++] = new_expense;
    printf("Expense added successfully!\n");

    float total_expense = get_total_expense(tracker);
    if (total_expense > tracker->budget) {
        printf("!!! Alert: Total expense %.2f exceeds the budget %.2f !!!",total_expense,tracker->budget);
    }
}
```

2. **Viewing Expenses:**

This function lists all expenses, helping review their spending history.

```c
void viewExpenses(ExpenseTracker *tracker) {
    printf("\nExpenses:\n");
    for (int i = 0; i < tracker->expense_count; i++) {
        printf("Category: %s, Amount: %.2f, Date: %s\n",
               tracker->expenses[i].category,
               tracker->expenses[i].amount,
               tracker->expenses[i].date);
    }
}
```

3. **Monthly and Annual Reports:**

```c
void monthlyReport(ExpenseTracker *tracker) {
    char ask_budget[2];
    float total = 0.0;
    char month[3];
    char year[5];
    printf("Enter month (MM): ");
    scanf("%s", month);
    printf("Enter year (YYYY): ");
    scanf("%s", year);

    printf("\nMonthly Report for %s-%s:\n", year, month);
    for (int i = 0; i < tracker->expense_count; i++) {
        if (strncmp(tracker->expenses[i].date + 5, month, 2) == 0 &&
            strncmp(tracker->expenses[i].date, year, 4) == 0) {
            printf("Category: %s, Amount: %.2f\n",
                   tracker->expenses[i].category,
                   tracker->expenses[i].amount);
            total += tracker->expenses[i].amount;
        }
    }
    printf("Total Expenses: %.2f\n", total);
}
```

```c
void annualReport(ExpenseTracker *tracker) {
    char ask_budget[2];
    float total = 0.0;
    char year[5];
    printf("Enter year (YYYY): ");
    scanf("%s", year);

    printf("\nAnnual Report for %s:\n", year);
    for (int i = 0; i < tracker->expense_count; i++) {
        if (strncmp(tracker->expenses[i].date, year, 4) == 0) {
            printf("Category: %s, Amount: %.2f\n",
                   tracker->expenses[i].category,
                   tracker->expenses[i].amount);
            total += tracker->expenses[i].amount;
        }
    }
    printf("Total Expenses: %.2f\n", total);
}
```

These functions filter and display expenses based on the input or year.

4. **Setting a Budget:**

```c
void setBudget(ExpenseTracker *tracker) {
    printf("Enter budget: ");
    scanf("%f", &tracker->budget);
    printf("Budget set to %.2f\n", tracker->budget);
}
```

Users can define or update their budget, which helps in monitoring overspending.

5. **Calculating Total Expenses:**

```c
float get_total_expense(ExpenseTracker *tracker) {
    float total = 0.0;
    for (int i = 0; i < tracker->expense_count; i++) {
        total += tracker->expenses[i].amount;
    }
    return total;
}
```

Users can define or update their budget, which helps in monitoring overspending.

RUNNING THE MAIN PROGRAM:

The main function initializes the tracker, sets a budget, and runs a menu-driven loop to perform different actions based on user input.

```c
int main() {
    ExpenseTracker tracker = { .expense_count = 0, .budget = 0.0 };
    int choice;

    setBudget(&tracker);
    do {
        displayMenu();
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1: addExpense(&tracker); break;
            case 2: viewExpenses(&tracker); break;
            case 3: monthlyReport(&tracker); break;
            case 4: annualReport(&tracker); break;
            case 5: setBudget(&tracker); break;
            case 0: printf("Exiting...\n"); break;
            default: printf("Invalid choice! Please try again.\n");
        }
    } while (choice != 0);

    return 0;
}
```

CONCLUSION

This C-based Expense Tracker offers a simple and practical solution to managing personal finances. While basic, it can be extended with features like saving data to files or adding graphical reports. The program emphasizes core C concepts like structures, loops, and function calls, making it a great project for learners.

Happy coding and budgeting!