UPES
UNIVERSITY OF TOMORROW

Subject: PROGRAMMING IN C

PROJECT

BTech CSE 1st Year

TEAM MEMBERS

Name: YASH NIRWAL & ABAAN ALI
SAP ID: 590022294 & 590024041
Batch: 68
Semester: 1ST
Date of submission: 29-11-2025

Faculty:
VINOD KUMAR

Bank Login & Passbook Managing System (C Language)

A comprehensive project report detailing the development of an automated banking system designed to streamline customer account management, transaction processing, and data handling through efficient file management and algorithmic design.

Problem Definition

Banking institutions maintain customer details including personal information, account numbers, transaction history, and account closure records. Traditionally, these details are written and updated manually, which increases chances of human error, slow processing, and data loss.

The objective of this project is to develop a Bank Login and Passbook Managing System in C that can:

- Create new bank accounts
- Generate unique 11-digit account numbers and 4-digit PIN numbers automatically
- Maintain account records using file handling
- Allow deposit, withdrawal, balance inquiry, and passbook viewing
- Allow account closure
- Keep a transaction history for the user

This automated system improves efficiency, accuracy, and reliability in handling basic banking operations.

Algorithm for Bank Login Page

01

Start Program

Initialize the banking system

02

Display Menu

Present options: Create New Account, Accounts, Close Account, Exit

03

Read User Choice

Accept input from user

04

Call Respective Function

Execute the selected operation

Algorithm for Creating New Account

1. **Display "New Account" Heading**

   Present the account creation interface to the user

2. **Request User Information**

   Ask user for Name, Date of Birth, and National ID Number

3. **Generate Unique Identifiers**

   Create new 11-digit account number (incremented) and new 4-digit PIN (incremented)

4. **Store Account Details**

   Save the new account details

5. **Display Confirmation**

   Show newly generated account number and PIN to user

6. **Return to Menu**

   Navigate back to Bank Login Menu

Algorithm for Bank Passbook Managing System

The Accounts menu provides comprehensive transaction management capabilities:

Deposit

Input amount, add to balance, and save in passbook.txt

Withdraw

Input amount, check if balance is sufficient, then deduct and record

Check Balance

Read last balance from file and print to user

Print Passbook

Display all transactions in chronological order

Additional features include Search Transaction by Date to match date and print results, and Back option to return to Bank Login.

Algorithm for Closing an Account

Account Closure Process

1. Display account closure screen
1. Ask user for Name, Date of Birth, and National ID Number
1. Read all accounts from accounts.txt
1. Write all accounts to a temporary file except the one to delete
1. Replace original file with updated file

Confirmation & Return

Display "Account Deleted Successfully" or "Account Not Found" message to user

Return to Bank Login menu after completion

Problems Faced by Our Group

File Handling Complexity

Ensuring accuracy while appending and modifying text files was challenging.

Auto-Generation of Account Number and PIN

Required scanning the last entry and generating the next number.

Passbook Formatting Issues

Aligning date, type, amount, and balance into columns needed careful formatting.

Account Deletion Logic

Required use of a temporary file to rewrite data safely.

Input Handling

Preventing unwanted newline characters from interrupting scanf operations.

Maintaining Data Consistency

Syncing passbook transactions with account details required attention.

Assumptions & Limitations

→ **Shared Passbook File**

Each account shares a common passbook file (passbook.txt) in this version.

→ **No PIN Validation**

PIN validation is not implemented for accessing the Accounts menu.

→ **Input Assumptions**

User inputs are assumed to be correct with no wrong-format validation.

→ **No Data Encryption**

The system does not encrypt sensitive data.

→ **Console-Based Interface**

The project is console-based and does not include GUI.

→ **Transaction History Retention**

Closing an account does not remove its past transactions.

→ **Date Format Compliance**

Date input is assumed to follow the correct format.

System Architecture Overview

Core Components

- Account Management Module
- Transaction Processing Engine
- File Handling System
- User Authentication Interface
- Data Validation Layer

Key Features

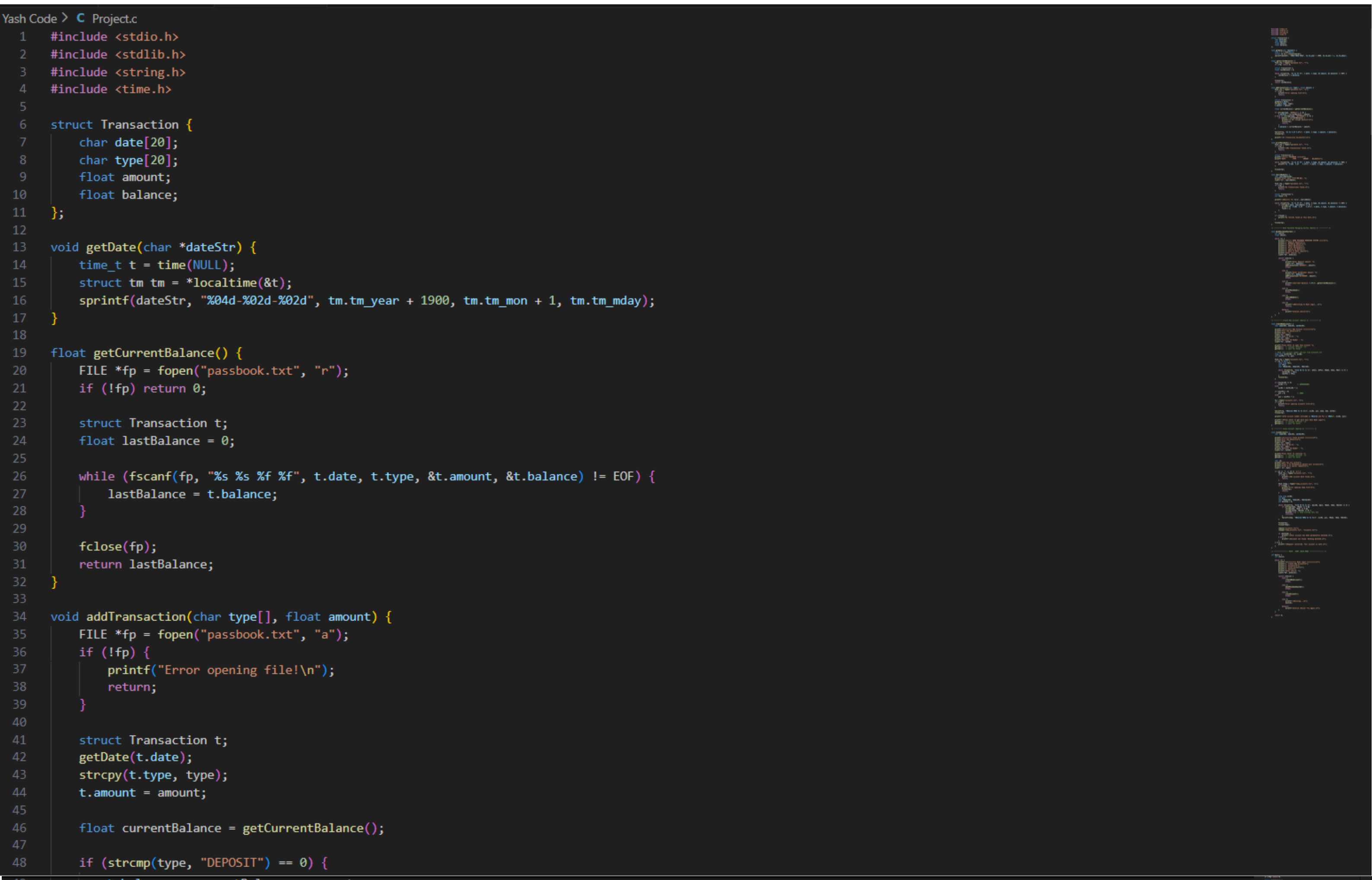
- Automatic account number generation
- Secure PIN creation
- Transaction history tracking
- Balance management
- Account closure functionality

Project Summary & Conclusion

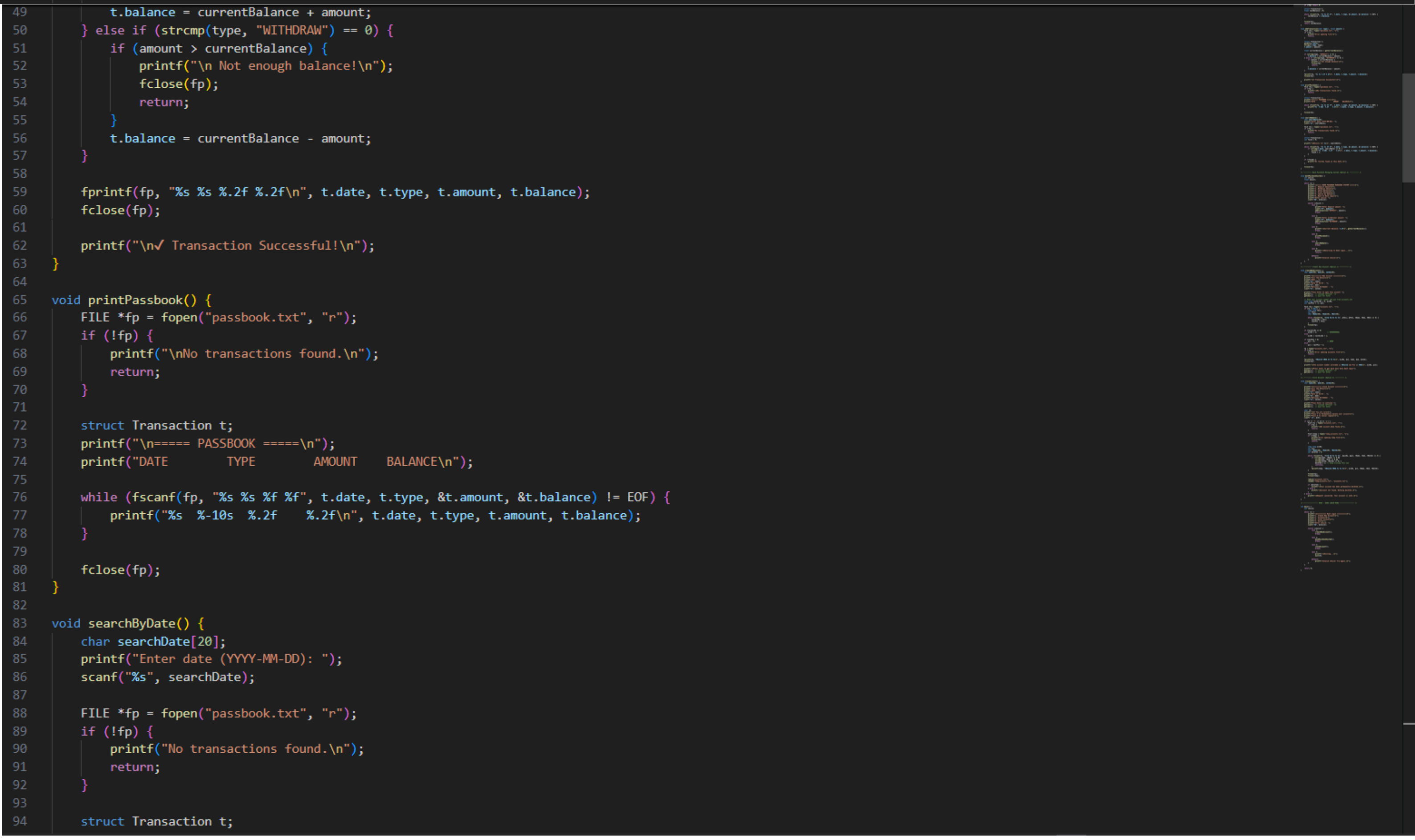
The Bank Login and Passbook Managing System in C represents a comprehensive solution to modernize banking operations by replacing manual record-keeping with an automated, file-based system. The project successfully addresses the core banking requirements of account creation, transaction management, and account closure while maintaining transaction history.

Key Achievement: This system demonstrates the practical application of C programming concepts including file handling, data structures, and algorithmic design to solve real-world banking challenges. Despite its limitations, the project provides a solid foundation for understanding how banking systems manage customer data and transactions efficiently.

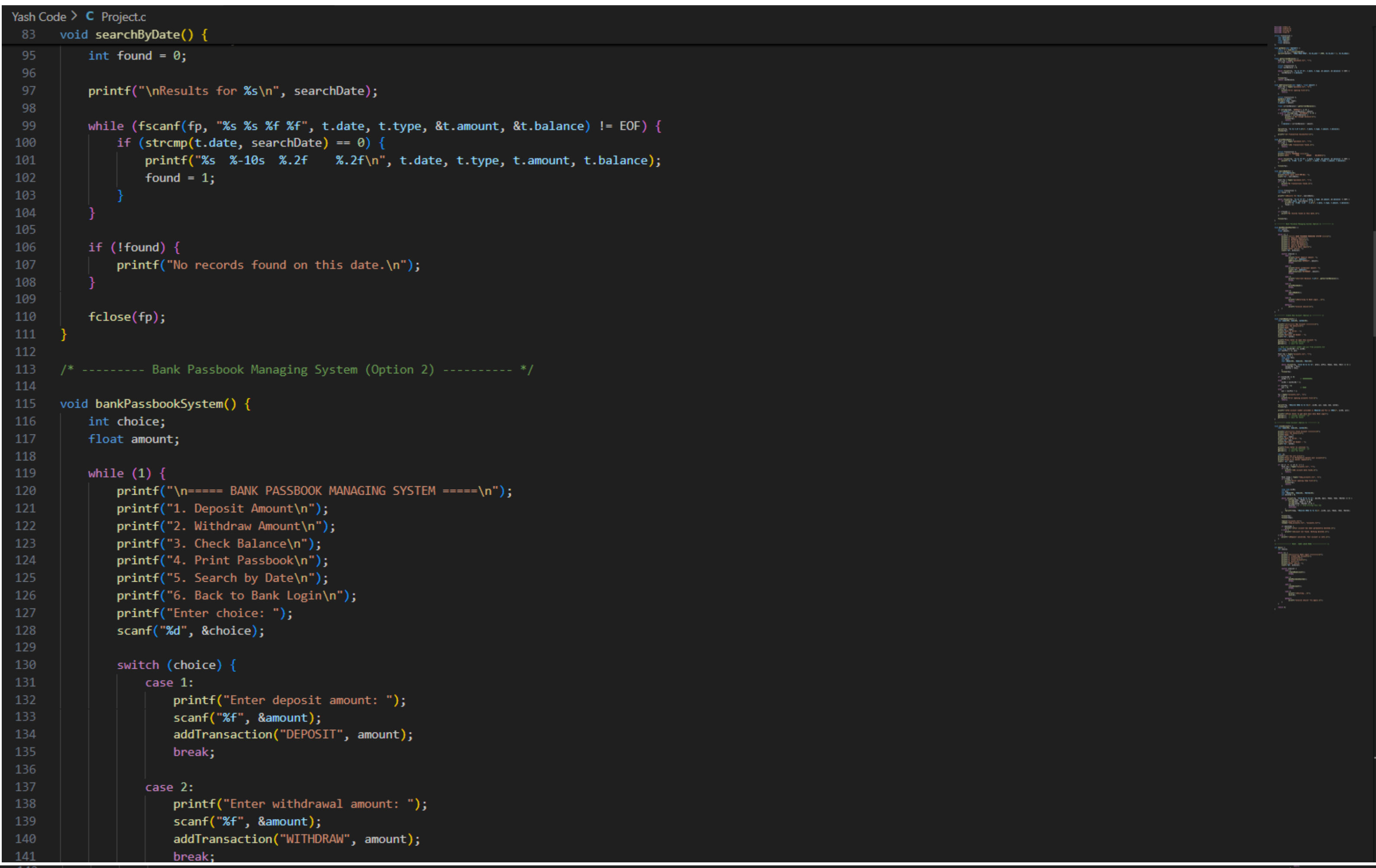
CODE OF THE PROJECT



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

struct Transaction {
    char date[20];
    char type[20];
    float amount;
    float balance;
};

void getDate(char *dateStr) {
    time_t t = time(NULL);
    struct tm tm = *localtime(&t);
    sprintf(dateStr, "%04d-%02d-%02d", tm.tm_year + 1900, tm.tm_mon + 1, tm.tm_mday);
}

float getCurrentBalance() {
    FILE *fp = fopen("passbook.txt", "r");
    if (!fp) return 0;

    struct Transaction t;
    float lastBalance = 0;

    while (fscanf(fp, "%s %s %f %f", t.date, t.type, &t.amount, &t.balance) != EOF) {
        lastBalance = t.balance;
    }

    fclose(fp);
    return lastBalance;
}

void addTransaction(char type[], float amount) {
    FILE *fp = fopen("passbook.txt", "a");
    if (!fp) {
        printf("Error opening file!\n");
        return;
    }

    struct Transaction t;
    getDate(t.date);
    strcpy(t.type, type);
    t.amount = amount;

    float currentBalance = getCurrentBalance();

    if (strcmp(type, "DEPOSIT") == 0) {
```

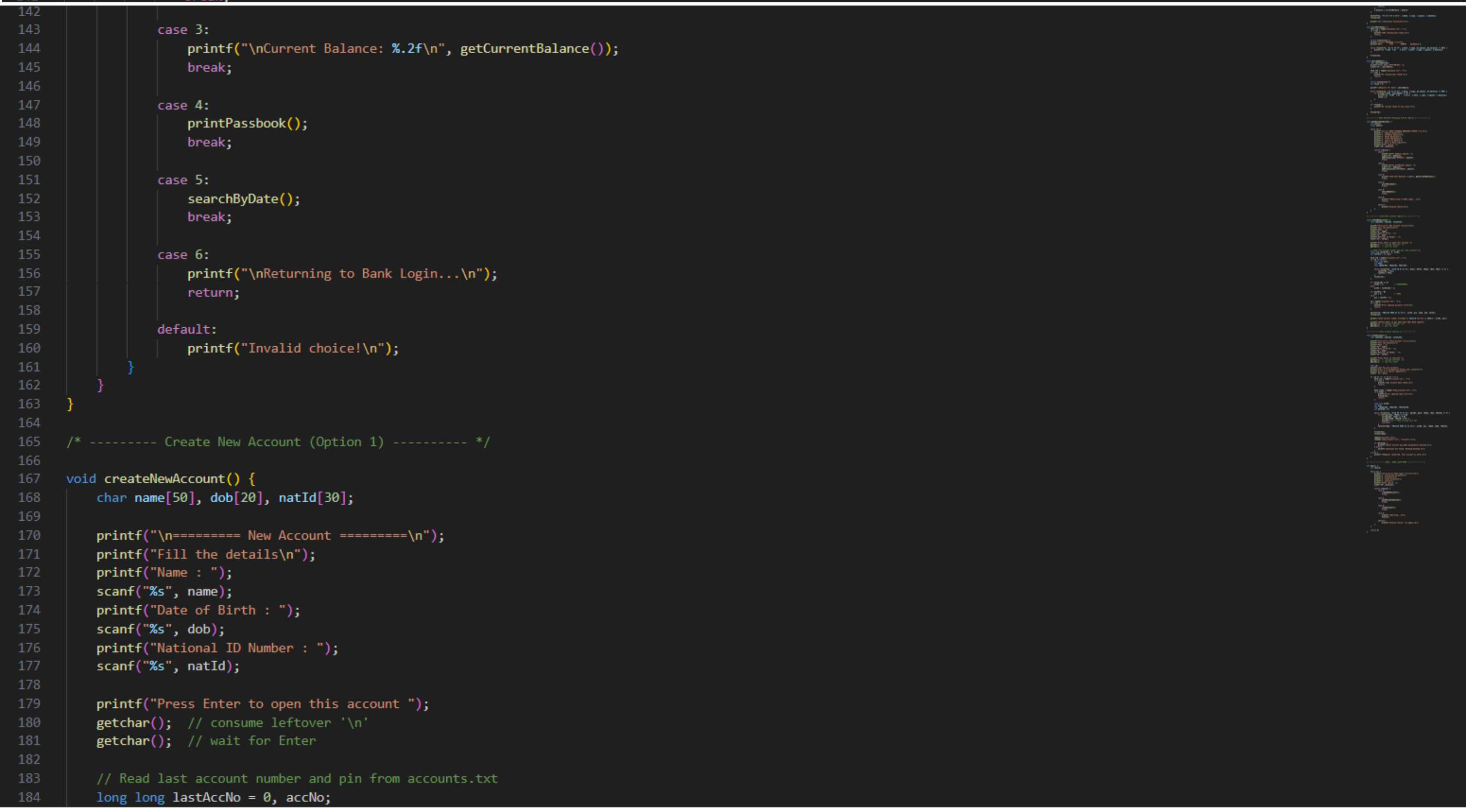


```c
        t.balance = currentBalance + amount;
    } else if (strcmp(type, "WITHDRAW") == 0) {
        if (amount > currentBalance) {
            printf("\n Not enough balance!\n");
            fclose(fp);
            return;
        }
        t.balance = currentBalance - amount;
    }

    fprintf(fp, "%s %s %.2f %.2f\n", t.date, t.type, t.amount, t.balance);
    fclose(fp);

    printf("\n✓ Transaction Successful!\n");
}

void printPassbook() {
    FILE *fp = fopen("passbook.txt", "r");
    if (!fp) {
        printf("\nNo transactions found.\n");
        return;
    }

    struct Transaction t;
    printf("\n===== PASSBOOK =====\n");
    printf("DATE        TYPE        AMOUNT    BALANCE\n");

    while (fscanf(fp, "%s %s %f %f", t.date, t.type, &t.amount, &t.balance) != EOF) {
        printf("%s  %-10s  %.2f    %.2f\n", t.date, t.type, t.amount, t.balance);
    }

    fclose(fp);
}

void searchByDate() {
    char searchDate[20];
    printf("Enter date (YYYY-MM-DD): ");
    scanf("%s", searchDate);

    FILE *fp = fopen("passbook.txt", "r");
    if (!fp) {
        printf("No transactions found.\n");
        return;
    }

    struct Transaction t;
```

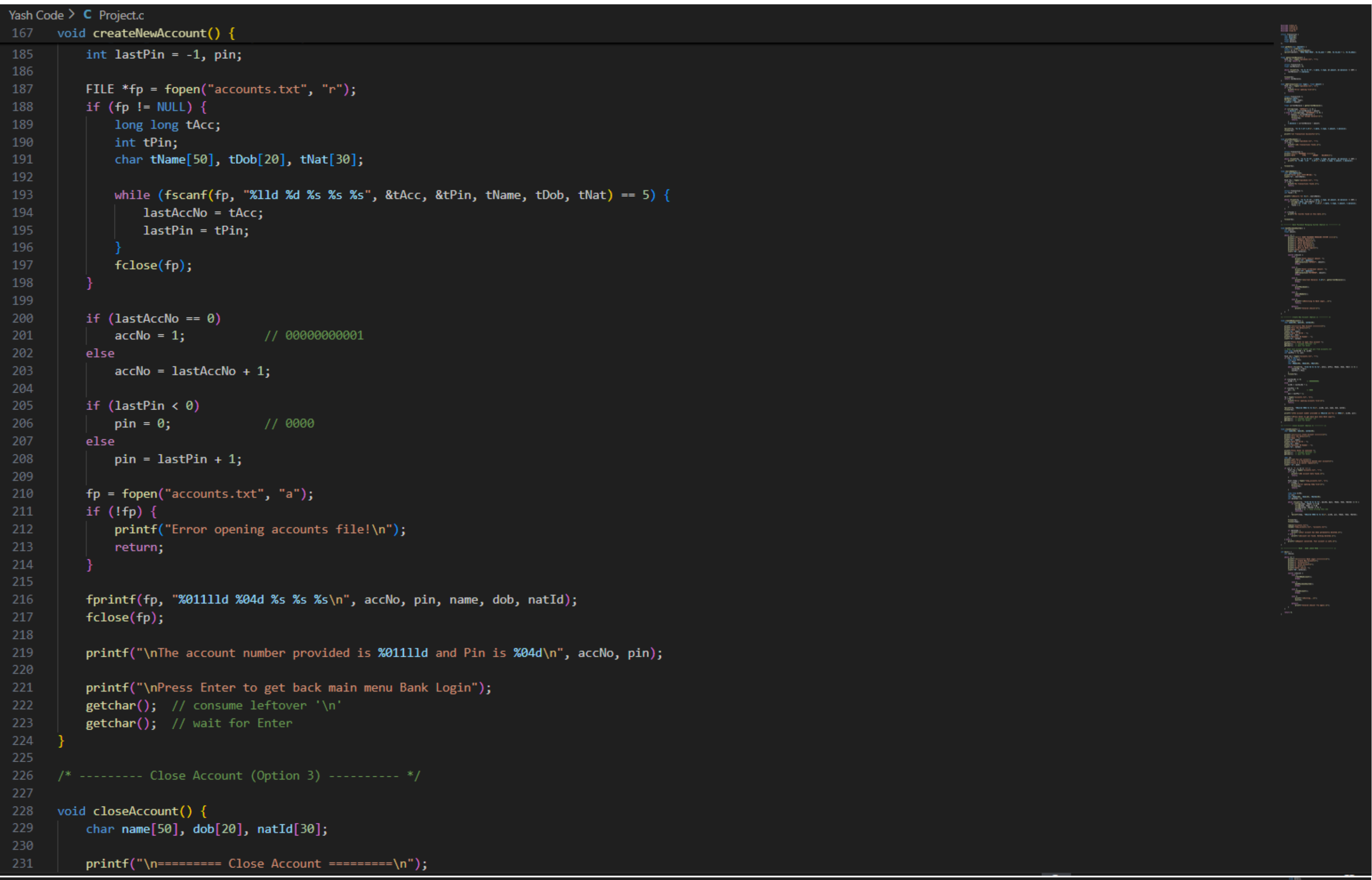


Yash Code > C Project.c

```c
void searchByDate() {
    int found = 0;

    printf("\nResults for %s\n", searchDate);

    while (fscanf(fp, "%s %s %f %f", t.date, t.type, &t.amount, &t.balance) != EOF) {
        if (strcmp(t.date, searchDate) == 0) {
            printf("%s  %-10s  %.2f    %.2f\n", t.date, t.type, t.amount, t.balance);
            found = 1;
        }
    }

    if (!found) {
        printf("No records found on this date.\n");
    }

    fclose(fp);
}

/* --------- Bank Passbook Managing System (Option 2) ---------- */

void bankPassbookSystem() {
    int choice;
    float amount;

    while (1) {
        printf("\n===== BANK PASSBOOK MANAGING SYSTEM =====\n");
        printf("1. Deposit Amount\n");
        printf("2. Withdraw Amount\n");
        printf("3. Check Balance\n");
        printf("4. Print Passbook\n");
        printf("5. Search by Date\n");
        printf("6. Back to Bank Login\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                printf("Enter deposit amount: ");
                scanf("%f", &amount);
                addTransaction("DEPOSIT", amount);
                break;

            case 2:
                printf("Enter withdrawal amount: ");
                scanf("%f", &amount);
                addTransaction("WITHDRAW", amount);
                break;
```

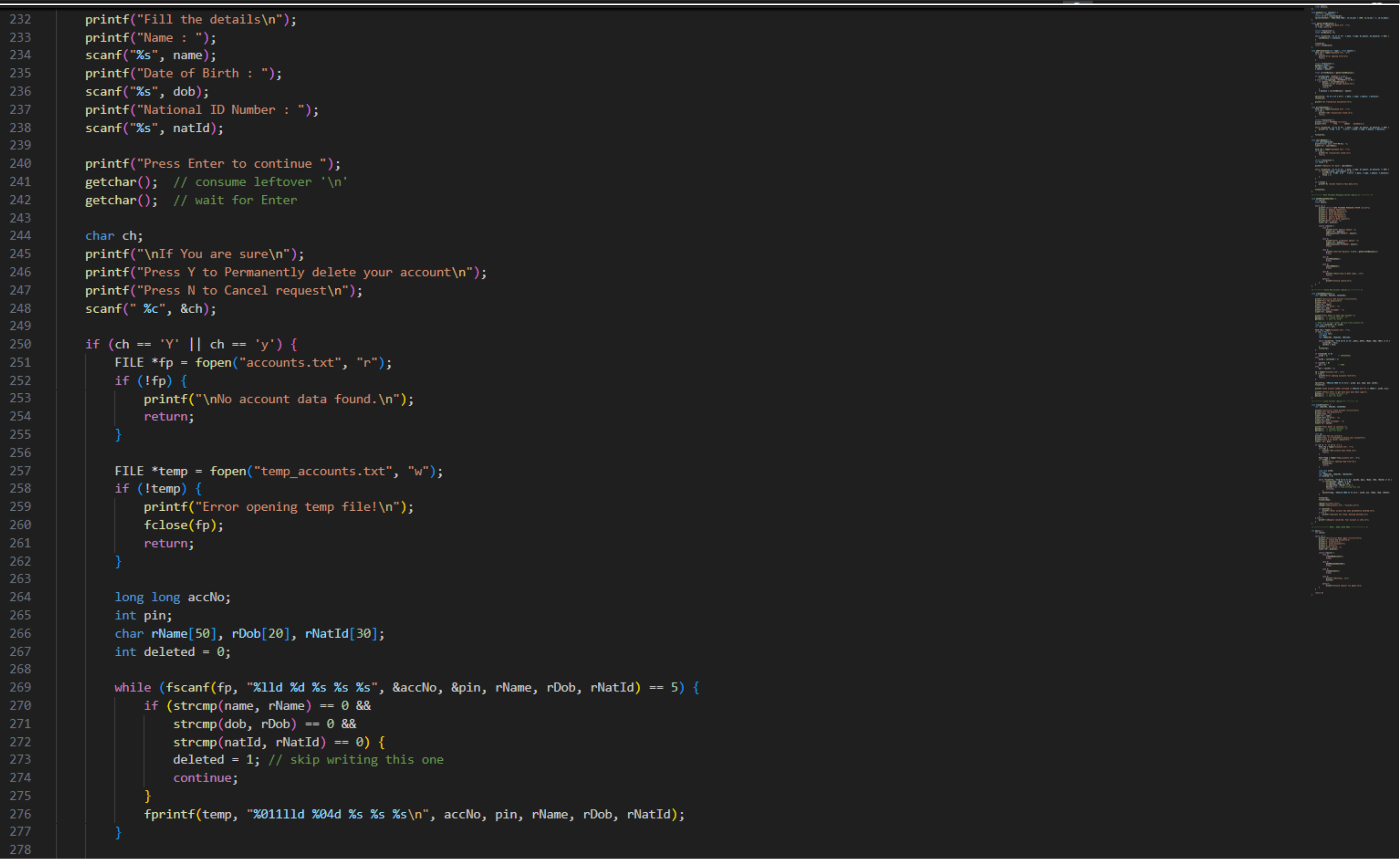


```c

            case 3:
                printf("\nCurrent Balance: %.2f\n", getCurrentBalance());
                break;

            case 4:
                printPassbook();
                break;

            case 5:
                searchByDate();
                break;

            case 6:
                printf("\nReturning to Bank Login...\n");
                return;

            default:
                printf("Invalid choice!\n");
        }
    }
}

/* --------- Create New Account (Option 1) ---------- */

void createNewAccount() {
    char name[50], dob[20], natId[30];

    printf("\n========= New Account =========\n");
    printf("Fill the details\n");
    printf("Name : ");
    scanf("%s", name);
    printf("Date of Birth : ");
    scanf("%s", dob);
    printf("National ID Number : ");
    scanf("%s", natId);

    printf("Press Enter to open this account ");
    getchar();  // consume leftover '\n'
    getchar();  // wait for Enter

    // Read last account number and pin from accounts.txt
    long long lastAccNo = 0, accNo;
```

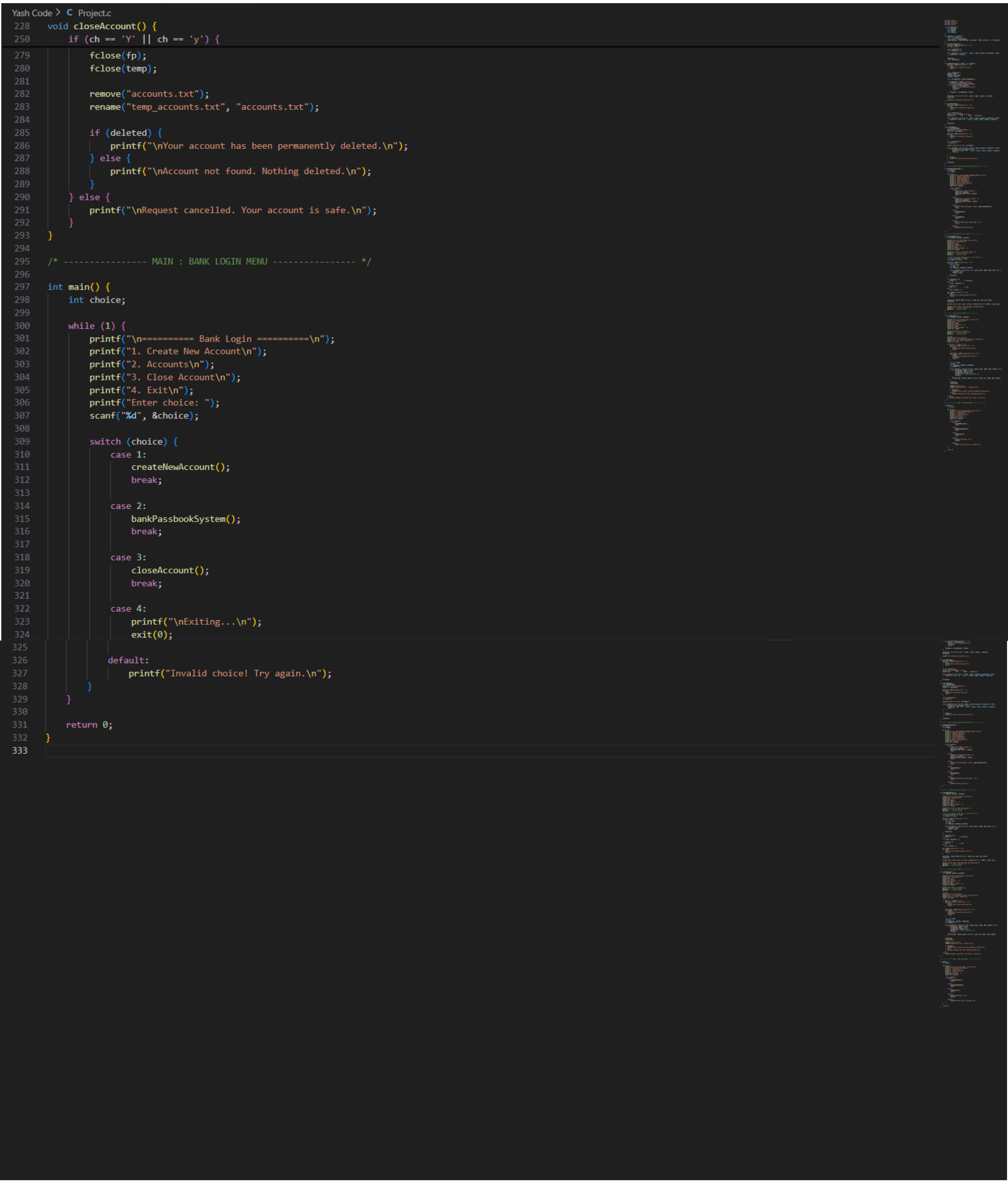


Yash Code > C Project.c

```c
void createNewAccount() {
    int lastPin = -1, pin;

    FILE *fp = fopen("accounts.txt", "r");
    if (fp != NULL) {
        long long tAcc;
        int tPin;
        char tName[50], tDob[20], tNat[30];

        while (fscanf(fp, "%lld %d %s %s %s", &tAcc, &tPin, tName, tDob, tNat) == 5) {
            lastAccNo = tAcc;
            lastPin = tPin;
        }
        fclose(fp);
    }

    if (lastAccNo == 0)
        accNo = 1;          // 00000000001
    else
        accNo = lastAccNo + 1;

    if (lastPin < 0)
        pin = 0;            // 0000
    else
        pin = lastPin + 1;

    fp = fopen("accounts.txt", "a");
    if (!fp) {
        printf("Error opening accounts file!\n");
        return;
    }

    fprintf(fp, "%011lld %04d %s %s %s\n", accNo, pin, name, dob, natId);
    fclose(fp);

    printf("\nThe account number provided is %011lld and Pin is %04d\n", accNo, pin);

    printf("\nPress Enter to get back main menu Bank Login");
    getchar();  // consume leftover '\n'
    getchar();  // wait for Enter
}

/* --------- Close Account (Option 3) ---------- */

void closeAccount() {
    char name[50], dob[20], natId[30];

    printf("\n========= Close Account =========\n");
```



```c
    printf("Fill the details\n");
    printf("Name : ");
    scanf("%s", name);
    printf("Date of Birth : ");
    scanf("%s", dob);
    printf("National ID Number : ");
    scanf("%s", natId);

    printf("Press Enter to continue ");
    getchar();  // consume leftover '\n'
    getchar();  // wait for Enter

    char ch;
    printf("\nIf You are sure\n");
    printf("Press Y to Permanently delete your account\n");
    printf("Press N to Cancel request\n");
    scanf(" %c", &ch);

    if (ch == 'Y' || ch == 'y') {
        FILE *fp = fopen("accounts.txt", "r");
        if (!fp) {
            printf("\nNo account data found.\n");
            return;
        }

        FILE *temp = fopen("temp_accounts.txt", "w");
        if (!temp) {
            printf("Error opening temp file!\n");
            fclose(fp);
            return;
        }

        long long accNo;
        int pin;
        char rName[50], rDob[20], rNatId[30];
        int deleted = 0;

        while (fscanf(fp, "%lld %d %s %s %s", &accNo, &pin, rName, rDob, rNatId) == 5) {
            if (strcmp(name, rName) == 0 &&
                strcmp(dob, rDob) == 0 &&
                strcmp(natId, rNatId) == 0) {
                deleted = 1; // skip writing this one
                continue;
            }
            fprintf(temp, "%011lld %04d %s %s %s\n", accNo, pin, rName, rDob, rNatId);
        }

```


Yash Code > C Project.c

```c
void closeAccount() {
    if (ch == 'Y' || ch == 'y') {
        fclose(fp);
        fclose(temp);

        remove("accounts.txt");
        rename("temp_accounts.txt", "accounts.txt");

        if (deleted) {
            printf("\nYour account has been permanently deleted.\n");
        } else {
            printf("\nAccount not found. Nothing deleted.\n");
        }
    } else {
        printf("\nRequest cancelled. Your account is safe.\n");
    }
}

/* ---------------- MAIN : BANK LOGIN MENU ---------------- */

int main() {
    int choice;

    while (1) {
        printf("\n========== Bank Login ==========\n");
        printf("1. Create New Account\n");
        printf("2. Accounts\n");
        printf("3. Close Account\n");
        printf("4. Exit\n");
        printf("Enter choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createNewAccount();
                break;

            case 2:
                bankPassbookSystem();
                break;

            case 3:
                closeAccount();
                break;

            case 4:
                printf("\nExiting...\n");
                exit(0);

            default:
                printf("Invalid choice! Try again.\n");
        }
    }

    return 0;
}
```