Subject name - programming for problem solving using C

Subject Code - ESC103(Pr.)

Source Code:

```
    scanf("%d", &accountNumber);
    printf("Enter amount to deposit: ");
    scanf("%f", &amount);

    int i;
    for (i = 0; i < numAccounts; i++) {
        if (accounts[i].accountNumber == accountNumber) {
            accounts[i].balance += amount;
            printf("Deposit successful.\n");
            return;
        }
    }

    printf("Account not found.\n");
}

// Function to withdraw money from an account
void withdraw(struct Account accounts[], int numAccounts) {
    int accountNumber;
    float amount;
    printf("Enter account number: ");
    scanf("%d", &accountNumber);
    printf("Enter amount to withdraw: ");
    scanf("%f", &amount);

    int i;
    for (i = 0; i < numAccounts; i++) {
        if (accounts[i].accountNumber == accountNumber) {
            if (accounts[i].balance >= amount) {
                accounts[i].balance -= amount;
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Rabi Dutta\Documents\Sudarsan_Datta_D_41_C_Project_1.exe
- Output Size: 131.1435546875 KiB
- Compilation Time: 0.31s
```

```
                printf("Withdrawal successful.\n");
            } else {
                printf("Insufficient balance.\n");
            }
            return;
        }
    }

    printf("Account not found.\n");
}

// Function to view account balance
void viewBalance(struct Account accounts[], int numAccounts) {
    int accountNumber;
    printf("Enter account number: ");
    scanf("%d", &accountNumber);

    int i;
    for (i = 0; i < numAccounts; i++) {
        if (accounts[i].accountNumber == accountNumber) {
            printf("Account balance: %.2f\n", accounts[i].balance);
            return;
        }
    }

    printf("Account not found.\n");
}

int main() {
    struct Account accounts[100];
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Rabi Dutta\Documents\Sudarsan_Datta_D_41_C_Project_1.exe
- Output Size: 131.1435546875 KiB
- Compilation Time: 0.31s
```

Sel: 0 | Lines: 132 | Length: 3521 | Insert | Done parsing in 0.016 seconds

```c
    int numAccounts = 0;
    int choice;

    do {
        printf("Bank Management System\n");
        printf("1. Create Account\n");
        printf("2. Deposit\n");
        printf("3. Withdraw\n");
        printf("4. View Balance\n");
        printf("0. Exit\n");
        printf("Enter your choice: ");
        scanf("%d", &choice);

        switch (choice) {
            case 1:
                createAccount(accounts, &numAccounts);
                break;
            case 2:
                deposit(accounts, numAccounts);
                break;
            case 3:
                withdraw(accounts, numAccounts);
                break;
            case 4:
                viewBalance(accounts, numAccounts);
                break;
            case 0:
                printf("Exiting...\n");
                break;
            default:
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Rabi Dutta\Documents\Sudarsan_Datta_D_41_C_Project_1.exe
- Output Size: 131.1435546875 KiB
- Compilation Time: 0.31s
```

Sel: 0 | Lines: 132 | Length: 3521 | Insert | Done parsing in 0.016 seconds

```c
                printf("Invalid choice.\n");
                break;
        }

        printf("\n");
    } while (choice != 0);

    return 0;
}
```

Compile Log | Debug | Find Results | Close

```
Compilation results...
--------
- Errors: 0
- Warnings: 0
- Output Filename: C:\Users\Rabi Dutta\Documents\Sudarsan_Datta_D_41_C_Project_1.exe
- Output Size: 131.1435546875 KiB
- Compilation Time: 0.31s
```

Sel: 0 | Lines: 132 | Length: 3521 | Insert | Done parsing in 0.016 seconds

Output:

```
Bank Management System
1. Create Account
2. Deposit
3. Withdraw
4. View Balance
0. Exit
Enter your choice: 1
Enter account number: 12022002016018
Enter initial balance: 100000
Account created successfully.

Bank Management System
1. Create Account
2. Deposit
3. Withdraw
4. View Balance
0. Exit
Enter your choice: 4
Enter account number: 12022002016018
Account balance: 100000.00

Bank Management System
1. Create Account
2. Deposit
3. Withdraw
4. View Balance
0. Exit
Enter your choice: 0
```

Variable Description:

`struct Account accounts[100]` : An array of `Account` structures to store customer account information. It has a maximum capacity of 100 accounts.

`int numAccounts`: The number of existing accounts currently stored in the `accounts` array.

`int choice`: The user's choice of operation in the main menu.

`struct Account`: A structure representing a customer account, consisting of an `accountNumber` of type `int` and a `balance` of type `float`.

`void createAccount(struct Account accounts[], int *numAccounts)` : A function that creates a new account. It takes the `accounts` array and a pointer to `numAccounts` as parameters.

`void deposit(struct Account accounts[], int numAccounts)` : A function that allows the user to deposit money into an account. It takes the `accounts` array and `numAccounts` as parameters.

`void withdraw(struct Account accounts[], int numAccounts)` : A function that enables the user to withdraw money from an account. It takes the `accounts` array and `numAccounts` as parameters.

`void viewBalance(struct Account accounts[], int numAccounts)` : A function that displays the balance of a specific account. It takes the `accounts` array and `numAccounts` as parameters.

`do-while` loop: Executes a menu-driven program until the user chooses to exit.

`switch` statement: Determines the action to be performed based on the user's choice.

`printf`: Used to display output messages.

`scanf`: Used to accept user input.

File Description:

This is a C program for a simple Bank Management System. It uses a struct named "Account" to store customer account information which includes account number and balance. The program provides four basic banking operations: create a new account, deposit money into an account, withdraw money from an account, and view the account balance.

The "createAccount" function allows the user to create a new account and stores the account information in an array of "Account" structs. It also takes a pointer to an integer variable that holds the number of accounts currently stored in the array. If the number of accounts exceeds 100, the function prints an error message.

The "deposit" function allows the user to deposit money into an existing account. It takes the array of "Account" structs and the number of accounts stored in the array as input. The function prompts the user to enter the account number and the amount to deposit. It then searches the array for the account with the given account number and adds the deposited amount to its balance.

The "withdraw" function allows the user to withdraw money from an existing account. It takes the array of "Account" structs and the number of accounts stored in the array as input. The function prompts the user to enter the account number and the amount to withdraw. It then searches the array for the account with the given account number and subtracts the withdrawal amount from its balance if the balance is sufficient.

The "viewBalance" function allows the user to view the balance of an existing account. It takes the array of "Account" structs and the number of accounts stored in the array as input. The function prompts the user to enter the account number and searches the array for the account with the given account number. It then prints the balance of the account if it exists, otherwise, it prints an error message.

The "main" function is the starting point of the program. It initializes an array of "Account" structs and an integer variable to hold the number of accounts stored in the array. It then displays a menu of banking operations and prompts the user to select an operation. Based on the user's selection, it calls the corresponding function. The program continues to display the menu until the user chooses to exit.

Function Desciption:

The code provided includes four functions and a `main` function:

1. `void createAccount(struct Account accounts[], int *numAccounts)` : This function creates a new account by accepting an account number and initial balance from the user. It checks if there is space available in the `accounts` array and adds the new account to the array if possible.

2. `void deposit(struct Account accounts[], int numAccounts)` : This function allows the user to deposit money into an existing account. It prompts the user for the account number and amount to deposit. It searches for the account in the `accounts` array and updates the account's balance accordingly.

3. `void withdraw(struct Account accounts[], int numAccounts)` : This function enables the user to withdraw money from an existing account. It asks the user for the account number and amount to withdraw. It searches for the account in the `accounts` array, checks if there are sufficient funds, and updates the account's balance if possible.

4. `void viewBalance(struct Account accounts[], int numAccounts)` : This function displays the balance of a specific account. It prompts the user for the account number and searches for the account in the `accounts` array to retrieve and display its balance.

5. `int main()`: The main function acts as a menu-driven program that allows users to interact with the bank management system. It displays a menu with options to create an account, deposit, withdraw, view balance, or exit. Based on the user's choice, it calls the respective functions to perform the desired operation.

Note: The code assumes a maximum capacity of 100 accounts (`accounts[100]`) and keeps track of the number of existing accounts using the `numAccounts` variable.