Exp No: 1 Date: 9 sept 21

Aim: Write a C program to implement a Banking System application using array of structure variables.

Theory:

Structures: User defined data types.

DMA: Dynamic memory allocation is the process of assigning the memory space during the execution time or the run time.

 Pointer: A variable which stores the address of another variable. They are faster.

CODE:

#include <stdio.h>

struct bank //A structure with all necessary variables.

{

    int id, balance, transaction;

    char name[30];

} \* user;

int main()

{

    system("cls"); //clear console

    int TUsr;

    printf("Enter Total users:\n");

    scanf("%d", &TUsr);

    int n, j = 1;

    user = (struct bank \*)malloc(TUsr \* sizeof(struct bank)); //DMA

    user->transaction = NULL;

    user[1].transaction = 0; //for some reason this didn't output 0 without doing this

    while (j)

    {

        printf("BankMenu\n\n"); //Menu

        printf("1 Create Account\n");

        printf("2 Deposit\n");

        printf("3 Withdraw\n");

        printf("4 Checkbalance\n");

        printf("5 Exit\n");

        printf("Enter from options:\n");

        scanf("%d", &n); //What user want?

        switch (n)

        {

        case 1:

            CreateAccount(j);

            break;

        case 2:

            Deposit();

            break;

        case 3:

            Withdraw();

            break;

        case 4:

            CheckBalance();

            break;

        case 5:

            free(user);

            exit(0);

            break;

        default:

            printf("Invalid Input");

            break;

        }

        j++;

        printf("\n\n");

    }

    return 0;

}

void CreateAccount(int n) //creates account

{

    printf("Please Enter you first name:\n");

    scanf("%s", &user[n].name);

    int ok;

    printf("Input 1 to deposit 1000 rupees:\n");

    scanf("%d", &ok);

    if (ok == 1)

    {

        user[n].id = n; //assining id autometically

        user[n].balance = 1000;

        printf("Account created! Please remember your id!:\n");

        printf("Your name, id and balance are:\n%s\n%d\n%d", user[n].name, user[n].id, user[n].balance);

    }

    else

    {

        printf("Cancelled\n");

    }

    user[n].transaction++;

}

void Deposit() //deposit money

{

    int EnteredId = Authticate();

    if (EnteredId == 0)

    {

        return;

    }

    int EnteredAmount = 0;

    printf("Enter amount to deposit\n");

    scanf("%d", &EnteredAmount);

    user[EnteredId].balance += EnteredAmount;

    printf("\nBalance is %d\n", user[EnteredId].balance);

    user[EnteredId].transaction++;

}

void Withdraw() //withdraw money

{

    int EnteredId = Authticate();

    if (EnteredId == 0)

    {

        return;

    }

    int EnteredAmount = 0, CheckAmount = 0;

    printf("Enter amount to Withdraw\n");

    scanf("%d", &EnteredAmount);

    CheckAmount = user[EnteredId].balance - EnteredAmount;

    if (CheckAmount > 500)

    {

        user[EnteredId].balance -= EnteredAmount;

        printf("\nBalance is %d\n", user[EnteredId].balance);

    }

    else

    {

        printf("Transaction failed as balace goes below minimum amount");

    }

    user[EnteredId].transaction++;

}

void CheckBalance() //check account information and balance

{

    int EnteredId = Authticate();

    if (EnteredId == 0)

    {

        return;

    }

    printf("Account info:\nID: %d\nName: %s\nBalance: %d\nTotal transactions performed: %d", user[EnteredId].id, user[EnteredId].name, user[EnteredId].balance, user[EnteredId].transaction);

}

int Authticate() //check is user exist.

{

    int EnteredId = 0, EnteredAmount = 0;

    char EnteredName[30];

    printf("Enter id\n");

    scanf("%d", &EnteredId);

    if (user[EnteredId].id != EnteredId)

    {

        printf("id not found");

        return 0;

    }

    printf("Enter name\n");

    scanf("%s", &EnteredName);

    if (strcmp(user[EnteredId].name, EnteredName) != 0)

    {

        printf("Entered name doesn't match");

        return 0;

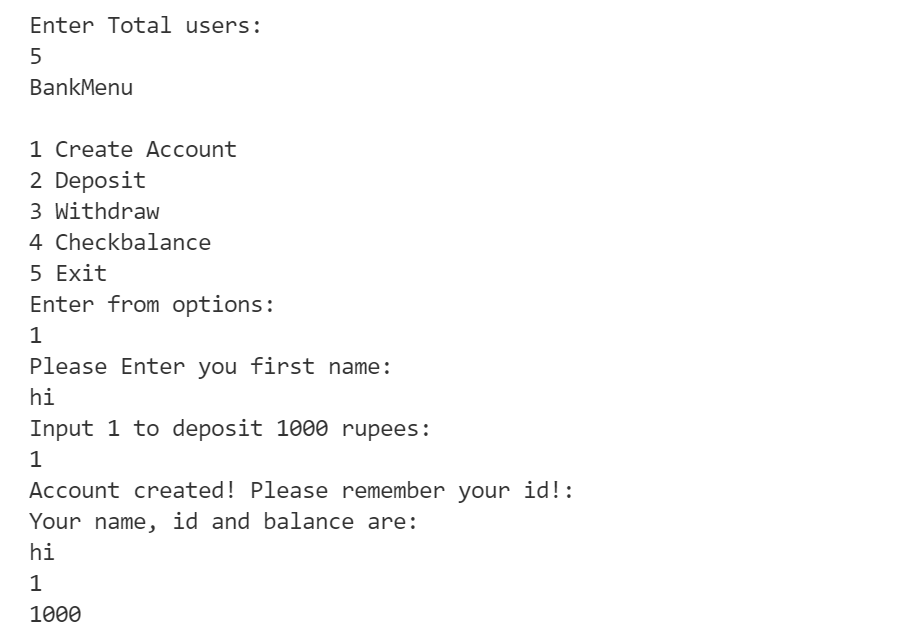
    }

    return EnteredId;

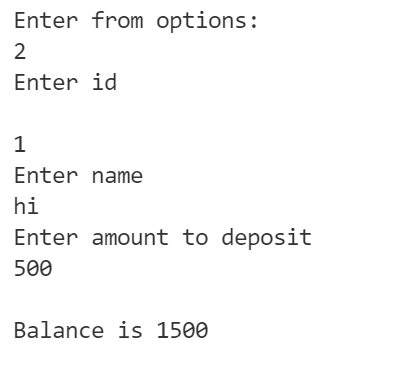
}

OUTPUT:

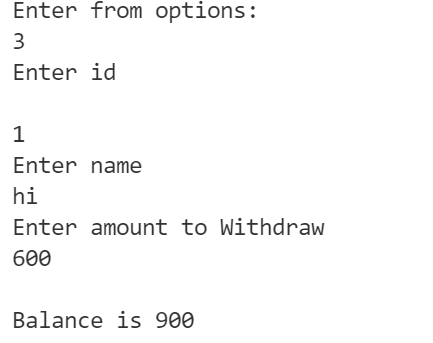
Creating account:



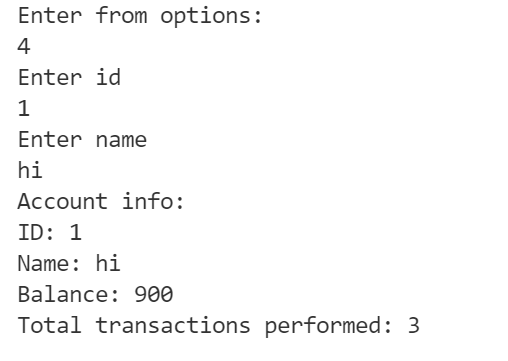
Deposit:



Withdraw



Check Balance:



Conclusion: Using structure makes work a lot easier. Using array would have created a mess.

DMA saves lot of space as we are only allotting required amount.