Experiment No. 4
Interactive ATM Application Using Looping Constructs in C

Aim: Create a simple ATM simulation program in C that allows users to perform basic banking operations such as checking account balance, depositing money, withdrawing money, and exiting the application. Use while and do-while loops to repeatedly display the menu and manage user interaction until the user chooses to exit. This application mirrors the functionality of a real ATM interface and demonstrates the use of iterative control structures in practical scenarios.

Learning Outcomes:
After completing this experiment, students will be able to:

- Understand and apply **control structures** such as loops and conditional statements in C programming.
- Develop **menu-driven applications** that enable continuous user interaction and perform operations based on user input.
- Strengthen **problem-solving and logical reasoning skills** by applying **input validation techniques** to create reliable and user-friendly programs.

Theory: The experiment demonstrates the use of fundamental C programming concepts, particularly control structures and user interaction.

The ATM Simulation Program demonstrates the practical use of control structures in the C programming language, particularly iterative (looping) and decision-making constructs. Control structures are essential in programming as they define the logical flow of execution within a program.

1. Control Structures in C: Control structures determine how instructions are executed in a program. They are classified into three main types:

a. Sequential Control:
The default mode where statements execute one after another in the order they appear.

b. Decision-Making (Selection) Control:
Used to make choices based on conditions. Examples include:
if, if-else, nested if, else-if ladder, and switch statements.
In this experiment, decision-making statements are used to select operations such as checking balance, depositing, or withdrawing money.

c. Iterative (Looping) Control:
Used to execute a block of code repeatedly until a certain condition is met. Examples include:

- while loop
- do-while loop
- for loop

In this experiment, while and do-while loops are used to repeatedly display the ATM menu and handle user input until the user decides to exit.

2. Iterative Control Structures

while loop:

Executes a block of code as long as a specified condition is true. The condition is checked before the loop executes.

Example:

```
while (choice != 4) {
   // display menu and process user input
}
```

do-while loop:

Executes a block of code at least once, and then repeats the loop as long as the condition remains true. The condition is checked after the loop executes.

Example:

```
do {
   // display menu and process user input
} while (choice != 4);
```

3. Decision-Making Structures

- if-else statements:

Used to check conditions such as whether sufficient balance is available before allowing withdrawal.

- switch statement:

Used to handle multiple menu options in a structured and readable way, simplifying the selection process based on user input.

4. Application Context: This experiment mirrors the functionality of a real ATM system, allowing users to:

- Check their account balance
- Deposit money
- Withdraw money
- Exit the application

It emphasizes user interaction, input validation, and logical control flow, showcasing how loops and conditionals can work together to build an interactive, menu-driven application.

Program Code:

Students are expected to implement this experiment by writing the complete C program for the Interactive ATM Application Using Looping Constructs in C.

Output:

Students are expected to execute the program and provide various sample outputs for the same code to demonstrate its functionality with different input values.

Conclusion: In this experiment, an ATM simulation program was successfully developed using loops and conditional statements in C. Through this practical task, the concept of control structures—both decision-making and iteration—was effectively applied to a real-world scenario, reinforcing the understanding of program flow, user interaction, and structured programming in C.