

# Bank Management System ✅

# Name: Kareem Mohamed Adel Khaled Department: ITCE Level: 3 Faculty: Faculty of Engineering, Suez Canal University Institute: ITI

## Introduction

This C++ program demonstrates a Bank Management System with multiple programming concepts including functions, conditionals, loops, object-oriented programming (OOP), and data structures like arrays, vectors, stacks, and queues.

## Wait for Enter (Function)

void waitEnter() {  
 cout << "\nPress Enter to continue...";  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
}

## Bank Greeting (Function)

void bankGreeting() {  
 cout << "=== Welcome to the Bank Management System ===\n";  
 cout << "Hello, valued customer!\n";  
 cout << "We are here to serve your banking needs.\n";

.

## Bank Data Calculations (Variables & Operators)

void bankDataCalculations() {  
 int a = 10;  
 double b = 3.5;  
 char c = 'X';  
 bool flag = true;  
 auto sum = a + b;  
 cout << "int a = " << a << "\n";  
 cout << "double b = " << b << "\n";  
 cout << "char c = " << c << "\n";  
 cout << "bool flag = " << (flag ? "true" : "false") << "\n";  
 cout << "a + b = " << sum << "\n";  
 cout << "a \* 2 = " << (a \* 2) << "\n";  
 cout << "b / 2 = " << (b / 2.0) << "\n";  
 cout << "a % 3 = " << (a % 3) << "\n";  
 cout << "++a -> " << (++a) << "\n";  
}

## Customer Status Check (Conditionals)

void customerStatusCheck() {  
 int num;  
 cout << "Enter an integer: ";  
 while(!(cin >> num)) {  
 cout << "Please enter a valid integer: ";  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
 }  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
 if (num > 0) cout << num << " is positive.\n";  
 else if (num < 0) cout << num << " is negative.\n";  
 else cout << num << " is zero.\n";  
 int last = abs(num) % 10;  
 switch(last) {  
 case 0: cout << "ends with 0\n"; break;  
 case 1: cout << "ends with 1\n"; break;  
 default: cout << "ends with other digit\n"; break;  
 }

## Transaction Processing Loop (Loops)

void transactionProcessingLoop() {  
 for (int i = 1; i <= 5; ++i) cout << i << " ";  
 int x = 5;  
 while (x > 0) { cout << x << " "; --x; }  
 int y;  
 do {  
 cout << "Enter 0 to stop: ";  
 while(!(cin >> y)) {  
 cin.clear();  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
 }  
 } while (y != 0);  
 cin.ignore(numeric\_limits<streamsize>::max(), '\n');  
}

## Banking Functions (Functions & Lambdas)

int add(int x, int y) { return x + y; }  
void bankingFunctions() {  
 cout << add(7,5) << "\n";  
 auto greet = [](const string &name = "Guest") {  
 cout << "Hello, " << name << "!\n";  
 };  
 greet("Kareem");  
 greet();  
}

## Account Management (OOP)

class Account {  
private:  
 string owner;  
 int accNumber;  
 double balance;  
public:  
 Account(string owner\_, int accNumber\_, double balance\_=0.0)  
 : owner(owner\_), accNumber(accNumber\_), balance(balance\_) {}  
 void deposit(double amount) { if (amount > 0) balance += amount; }  
 bool withdraw(double amount) {  
 if (amount > 0 && amount <= balance) { balance -= amount; return true; }  
 return false;  
 }  
 void print() const {  
 cout << "Account " << accNumber << " | Owner: " << owner  
 << " | Balance: $" << balance << "\n";  
 }  
};  
void accountManagement() {  
 Account a("Alice", 1001, 250.0);  
 a.deposit(50);  
 a.withdraw(100);  
 a.print();  
}

## Bank Records Storage (Arrays, Vector)

void bankRecordsStorage() {  
 int arr[5] = {1,2,3,4,5};  
 array<int, 4> a = {10, 20, 30, 40};  
 vector<string> customers = {"Alice","Bob","Charlie"};

.

## Customer Name Operations (String)

void customerNameOperations() {  
 string s1 = "Hello";  
 string s2 = "World";  
 string s3 = s1 + ", " + s2 + "!";  
 cout << s3 << "\n";  
}

## Transaction History (Stack)

void transactionHistory() {  
 stack<string> st;  
 st.push("Deposit $100");  
 st.push("Withdraw $50");  
 st.push("Deposit $200");

## Service Queue (Queue)

void serviceQueue() {  
 queue<string> q;  
 q.push("Customer1");  
 q.push("Customer2");  
}

## Explanation to code

* **<iostream>** – For input/output operations (cin, cout).
* **<vector>** – Dynamic arrays that can grow or shrink in size.
* **<string>** – For working with text strings.
* **<stack>** – Stack data structure (LIFO).
* **<queue>** – Queue data structure (FIFO).
* **<array>** – Fixed-size arrays.
* **<limits>** – To clear input buffer using numeric\_limits.

#waitEnter()

Pauses the program until the user presses Enter.  
This is used after each demo to give the user time to read the output before continuing.

**3. Demo Functions**

**3.1 Hello World**

Prints “Hello, world!” and demonstrates the most basic output statement using cout.

**3.2 Variables, Types, and Operators**

Shows how to declare and use different variable types:  
int, double, char, bool.  
Performs arithmetic operations and increments.

**3.3 Conditionals**

Reads an integer from the user, determines if it’s positive, negative, or zero using if/else.  
Also uses a switch statement to check the last digit of the number.

**3.4 Loops**

Demonstrates:

* for loop → counts from 1 to 5.
* while loop → counts down from 5 to 1.
* do-while loop → repeats until the user enters 0.

**3.5 Functions**

Defines a simple add() function to return the sum of two integers.  
Uses a **lambda function** to greet the user.

**3.6 Classes & Abstraction**

Defines an Account class with attributes:

* Owner name
* Account number
* Balance

Methods allow deposits, withdrawals, and printing account details.  
The demo creates an account and performs these operations.

**3.7 Arrays**

Shows:

* C-style arrays
* std::array
* std::vector

**3.8 Strings**

Demonstrates:

* Concatenation
* Finding string length
* Extracting substrings
* Searching for words in a string

**3.9 Stack**

Demonstrates **Last-In-First-Out**:

* Adding items with push()
* Viewing the last item with top()
* Removing items with pop()

**3.10 Queue**

Demonstrates **First-In-First-Out**:

* Adding items with push()
* Viewing first and last items with front() and back()
* Removing items with pop()

**4. Main Program Flow**

The main() function runs a **do-while loop** that:

1. Displays the menu of demos.
2. Reads the user’s choice.
3. Runs the matching demo using a switch statement.
4. Calls waitEnter() to pause before repeating.

The loop ends when the user chooses **0** to exit.

**Conclusion**

This program is a practical guide to learning C++ basics.  
It combines **interactive learning** with **hands-on coding examples**, making it suitable for beginners who want to see how different concepts work in action.