```cpp

#include <iostream>

#include <vector>

#include <string>

using namespace std;

```

- `#include <iostream>`: This line includes a C++ standard library header that provides basic input and output services, commonly used for tasks like reading from the standard input and writing to the standard output.

- `#include <vector>`: This line includes a header for using the `vector` container class template, which is a part of the C++ Standard Library. Vectors are dynamic arrays that can resize themselves automatically when elements are added or removed.

- `#include <string>`: This line includes a header for using the `string` class, which is a part of the C++ Standard Library. It provides a convenient way of working with strings of characters.

- `using namespace std;`: This line declares that elements from the C++ Standard Library (such as `cout`, `cin`, `vector`, and `string`) are used from the standard namespace `std`. This means you don't have to prefix these elements with `std::` every time you use them in the code.

```cpp

struct Account {

int account\_number;

string name;

float balance;

};

```

- `struct Account { ... };`: This defines a structure named `Account` that holds three pieces of data: an integer `account\_number`, a string `name`, and a floating-point number `balance`. A structure is a user-defined data type in C++ that allows you to group different variables under a single name.

```cpp

void create\_account(vector<Account> &accounts);

void deposit(vector<Account> &accounts);

void withdraw(vector<Account> &accounts);

void view\_account(const vector<Account> &accounts);

```

- `void create\_account(vector<Account> &accounts);`: This line declares a function named `create\_account` that takes a reference to a vector of `Account` objects as a parameter and does not return any value (`void` means the function does not return any value).

- `void deposit(vector<Account> &accounts);`: This line declares a function named `deposit` with a similar structure as `create\_account`.

- `void withdraw(vector<Account> &accounts);`: This line declares a function named `withdraw` with a similar structure as `create\_account`.

- `void view\_account(const vector<Account> &accounts);`: This line declares a function named `view\_account` with a similar structure as `create\_account`. The `const` keyword indicates that the function does not modify the passed `accounts` vector.

```cpp

int main() {

int choice;

vector<Account> accounts;

```

- `int main() { ... }`: This defines the main function, which is the entry point of every C++ program. The program execution starts from here.

- `int choice;`: This declares an integer variable named `choice` to store the user's menu selection.

- `vector<Account> accounts;`: This declares a vector named `accounts` that can hold objects of the `Account` structure. It is initially empty.

```cpp

while (true) {

cout << "\n\n\*\*\*BANK MANAGEMENT SYSTEM\*\*\*\n";

cout << "1. Create Account\n";

cout << "2. Deposit\n";

cout << "3. Withdraw\n";

cout << "4. View Account Details\n";

cout << "5. Exit\n";

cout << "Enter your choice: ";

cin >> choice;

```

- `while (true) { ... }`: This is an infinite loop that repeatedly displays the menu options to the user and waits for their choice.

- `cout << "..." << endl;`: The `cout` statement is used to print text to the standard output (usually the console). `<<` is the stream insertion operator, and `endl` is used to insert a newline character, making the output appear on a new line.

- `cin >> choice;`: The `cin` statement is used to read data from the standard input (usually the keyboard). `>>` is the stream extraction operator, used for input operations.

```cpp

switch (choice) {

case 1:

create\_account(accounts);

break;

case 2:

deposit(accounts);

break;

case 3:

withdraw(accounts);

break;

case 4:

view\_account(accounts);

break;

case 5:

cout << "\nThank you for using the banking system!\n";

exit(0);

default:

cout << "\nInvalid choice. Please try again.\n";

}

```

- `switch (choice) { ... }`: This is a switch statement that evaluates the value of `choice` and executes the corresponding case. If `choice` matches one of the cases (1, 2, 3, 4, or 5), the corresponding function (e.g., `create\_account`, `deposit`, etc.) is called.

- `case 1:`, `case 2:`, `case 3:`, `case 4:`, `case 5:`: These are labels within the switch statement. If `choice` matches one of these cases, the code following that case label is executed.

- `break;`: The `break` statement is used to exit the switch statement. After a `break` statement, the program flow moves outside the switch.

- `exit(0);`: The `exit` function is called to terminate the program with a status code of 0, indicating successful execution.

```cpp

void create\_account(vector<Account> &accounts) {

// Function implementation for creating a new account

}

```

- `void create\_account(vector<Account> &accounts) { ... }`: This line defines the `create\_account` function, which takes a reference to a vector of `Account` objects as a parameter and does not return any value (`void`).

- The function prompts the user for account number, checks if an account with the given number already exists, and if not, creates a new `Account` object and adds it to the `accounts` vector.

The other functions (`deposit`, `withdraw`, and `view\_account`) have similar structures where they take the `accounts` vector as a parameter, perform specific operations based on user input, and modify the account balances or display account details accordingly.