код программы:

```
#include <iostream>
#include <thread>
#include <mutex>

using namespace std;

class BankAccount {
private:
    int balance;
    mutex mtx;

public:
    BankAccount(int initialBalance) : balance(initialBalance) {}

    void takeMoney(int amount) {
        lock_guard<mutex> lock(mtx);
        if (balance >= amount) {
            balance -= amount;
            cout << "Thread " << this_thread::get_id() << " withdrew "
<< amount << " dollars. Balance: " << balance << endl;
        } else {
            cout << "Thread " << this_thread::get_id() << " tried to
withdraw " << amount << " dollars, but insufficient funds." << endl;
        }
    }

    int checkBalance() {
        return balance;
    }
};

class Client {
private:
    BankAccount& account;
    int amount;

public:
    Client(BankAccount& acc, int amt) : account(acc), amount(amt) {}

    void operator()() {
        while (account.checkBalance() > 0) {
            account.takeMoney(amount);
```

```
            this_thread::sleep_for(chrono::milliseconds(100)); // Pause
to simulate time taken to withdraw money
        }
    }
};

int main() {
    int initialBalance = 1000; // Initial balance
    const int numberOfClients = 5; // Number of clients
    int withdrawalAmount = 100; // Amount to withdraw by each client

    BankAccount account(initialBalance);

    cout << "Running with synchronization:" << endl;

    thread clients[numberOfClients];
    for (int i = 0; i < numberOfClients; ++i) {
        clients[i] = thread(Client(account, withdrawalAmount));
    }

    for (int i = 0; i < numberOfClients; ++i) {
        clients[i].join();
    }

    return 0;
}
```

2. Объяснение:
А) Класс BankAccount:

Описание: Представляет банковский счет с методами для снятия денег и проверки баланса.
Метод takeMoney: Синхронизирован с использованием lock_guard<mutex>, чтобы избежать конфликтов при одновременном доступе из разных потоков.
Логика: Если на счету достаточно денег, метод takeMoney снимает указанную сумму и выводит сообщение. Если средств недостаточно, выводится сообщение о недостатке средств.
Б) Класс Client:

Описание: Представляет клиента, который пытается снять деньги со счета.
Метод operator()(): Перегружен для использования объекта Client в потоках.
Логика: Клиент пытается снять деньги, пока на счету есть средства, с паузой в 100 миллисекунд между попытками.
В) Функция main:

Логика:
Создает объект BankAccount с начальным балансом.
Создает несколько потоков, каждый из которых представляет клиента, и запускает их.
Ожидает завершения работы всех потоков перед завершением работы программы.

результат: