Реферат програми для управління банківськими даними

Даний код реалізує програму для управління даними банку, включаючи роботу з клієнтами, їх вкладами та цінними паперами. Програма дозволяє завантажувати та зберігати інформацію у файли CSV, обробляти дані вручну, а також виконувати основні фінансові операції.

Основні функції програми

1. **Розрахунок прибутку по вкладу**
   Функція calculateProfit обчислює прибуток від депозиту, використовуючи формулу:

```
// Функція для розрахунку прибутку по вкладу
void calculateProfit(const Deposit& deposit) {
    double profit = deposit.amount * deposit.interest_rate * deposit.duration / 100;
    cout << "Прибуток по вкладу: " << profit << endl;
}
```

Прибутокп = Сума × Процентна ставка × Тривалість/100

   Вона виводить результат на екран.

2. **Робота з файлами CSV**
   - saveToCSV: Зберігає дані про клієнтів та їх депозити у файл у форматі CSV. Це дозволяє легко передавати дані в інші системи.

```
// Функція для збереження даних у CSV
void saveToCSV(const string& filename, const vector<Client>& clients) {
    ofstream file(filename);
    for (const auto& client : clients) {
        file << "Client name: " << quoted(client.name) << ","
             << client.balance << "\n";
        for (const auto& deposit : client.deposits) {
            file << quoted(deposit.type) << ","
                 << deposit.amount << ","
                 << deposit.interest_rate << ","
                 << deposit.duration << "\n";
        }
    }
}
```

   - loadFromCSV: Завантажує інформацію про клієнтів та їх депозити з файлу CSV. Вона використовує потокове зчитування та парсинг даних.

```
// Функція для завантаження даних з CSV
void loadFromCSV(const string& filename, vector<Client>& clients) {
    ifstream file(filename);
    string line;
    while (getline(file, line)) {
        stringstream ss(line);
        Client client;
        getline(ss, client.name, ',');
        ss >> client.balance;
        clients.push_back(client);
        while (getline(file, line)) {
            stringstream depositStream(line);
            Deposit deposit;
            getline(depositStream, deposit.type, ',');
            depositStream >> deposit.amount >> deposit.interest_rate >> deposit.duration;
            client.addDeposit(deposit);
        }
    }
}
```

3. **Виведення стану банку**
   Функція printBankState демонструє поточну інформацію про клієнтів, їх баланси та депозити у структурованому вигляді.

```
// Функція для виведення стану банку
void printBankState(const vector<Client>& clients) {
    for (const auto& client : clients) {
        client.displayClientInfo();
        cout << "==============================\n";
    }
}
```

4. **Оновлення та торгівля цінними паперами**
   - updateStockPrices: Збільшує ціни на цінні папери на заданий відсоток.

```
// Функція для зміни цін цінних паперів
void updateStockPrices(vector<Stock>& stocks, double percentIncrease) {
    for (auto& stock : stocks) {
        stock.price += stock.price * percentIncrease / 100;
    }
}
```

   - buyStock і sellStock: Додають або зменшують кількість цінних паперів у портфелі.

```
// Функція для покупки цінних паперів
void buyStock(vector<Stock>& stocks, const string& stockName, double price, int quantity) {
    stocks.push_back(Stock(stockName, price, quantity));
}

// Функція для продажу цінних паперів
void sellStock(vector<Stock>& stocks, const string& stockName, int quantity) {
    for (auto& stock : stocks) {
        if (stock.name == stockName && stock.quantity >= quantity) {
            stock.quantity -= quantity;
            break;
        }
    }
}
```

Класи та структури

1. **Deposit**
   Структура для представлення вкладів. Містить такі поля:
   - type: Тип вкладу (наприклад, строковий).
   - amount: Сума вкладу.
   - interest_rate: Процентна ставка.
   - duration: Тривалість вкладу (у роках).

```
// Структура для вкладу
struct Deposit {
    string type;          // Тип вкладу
    double amount;        // Сума
    double interest_rate; // Процентна ставка
    int duration;         // Строк (в роках)

    Deposit() : type("Терміновий"), amount(0), interest_rate(0), duration(0) {}
};
```

2. **Client**
   Клас для представлення клієнта банку. Містить:
   - name: Ім'я клієнта.
   - balance: Баланс клієнта.
   - deposits: Список депозитів.
     Методи:
   - addDeposit: Додає новий вклад до списку.
   - displayClientInfo: Виводить інформацію про клієнта та його вклади.

```
// Клас для клієнта
class Client {
public:
    string name;                // Ім'я клієнта
    double balance;             // Баланс клієнта
    vector<Deposit> deposits;   // Список вкладів

    Client(string n = "", double b = 0) : name(n), balance(b) {}

    void addDeposit(const Deposit& deposit) {
        deposits.push_back(deposit);
    }

    void displayClientInfo() const {
        cout << "\nКлієнт: " << name << "\nБаланс: " << balance << "\n=== Вклад ===\n";
        for (const auto& deposit : deposits) {
            cout << "\nТип: " << deposit.type << "\nСума: " << deposit.amount
                 << "\nПроцентна ставка: " << deposit.interest_rate << "%"
                 << "\nСтрок: " << deposit.duration << " років\n\n";
        }
    }
};
```

3. **Stock**
   Структура для опису цінних паперів. Параметри:
   - name: Назва цінного паперу.
   - price: Поточна ціна.
   - quantity: Кількість в обігу.

```cpp
// Структура для цінного паперу
struct Stock {
    string name;      // Ім'я цінного паперу
    double price;     // Поточна ціна
    int quantity;     // Кількість

    Stock(string n = "", double p = 0.0, int q = 0) : name(n), price(p), quantity(q) {}
};
```

Робота основної програми

1. **Інтерфейс користувача**
   Програма пропонує вибрати між завантаженням даних із файлу або введенням вручну. У разі ручного введення користувач додає клієнтів, їх депозити та цінні папери.

2. **Додавання клієнтів і вкладів**
   Програма дозволяє ввести інформацію про клієнтів, їх баланси, а також створювати депозити з відповідними параметрами.

3. **Додавання цінних паперів**
   Дозволяє створювати список цінних паперів із зазначенням їх назви, ціни та кількості.

4. **Оновлення цін цінних паперів**
   Користувач вводить відсоток збільшення, і програма оновлює ціни на основі цього значення.

5. **Розрахунок прибутку**
   Програма обчислює прибуток для першого депозиту першого клієнта (як приклад).

6. **Збереження даних**
   Дані про клієнтів та їх депозити можуть бути збережені у файл CSV, що забезпечує довготривале зберігання.

Особливості програми

- **Гнучкість введення даних:** Можливість завантажувати з файлів або вводити вручну.
- **Масштабованість:** Підтримує роботу з багатьма клієнтами та різними видами операцій.

- **Автоматизація фінансових обчислень:** Програма розраховує прибуток і оновлює ціни на цінні папери.

Висновки

Дана програма є універсальним інструментом для банківської діяльності. Вона дозволяє автоматизувати обробку даних про клієнтів, депозити та цінні папери, забезпечуючи ефективне управління банківськими ресурсами.