Doubly Linked List Implementation

Instructor: Bostan Khan

Doubly Linked List for Bank Account Management

- Objective: Efficiently manage bank accounts using a doubly linked list.
- Implementation Overview:
  - Utilizes a doubly linked list for account management.
  - Each node represents a bank account.
  - Features deposit, withdrawal, and account display functionalities.

Doubly Linked List for Bank Account Management

- Implementation Highlights:
    - Supports dynamic addition and removal of accounts.
    - Enables seamless deposit and withdrawal operations.
    - Essential for building robust banking systems.

Doubly Linked List for Bank Account Management

- Key takeaways:
  - Demonstrates practical application of data structures.
  - Crucial for developing efficient financial applications.
- Let's Explore Bank Account Management with Doubly Linked Lists!

The BankAccount Class

```cpp
// Example user-defined class: BankAccount
class BankAccount {
private:
    string accountNumber;
    double balance;

public:
    BankAccount(string accNum, double bal) : accountNumber(accNum), balance(bal) {}

    string getAccountNumber() const { return accountNumber; }
    double getBalance() const { return balance; }

    void deposit(double amount) { balance += amount; }
    void withdraw(double amount) { balance -= amount; }
};
```

The Node Class

```cpp
// Node class for the doubly linked list
class Node {
public:
    BankAccount account;
    Node* prev;
    Node* next;

    Node(BankAccount acc) : account(acc), prev(nullptr), next(nullptr) {}
};
```

The DoublyLinkedList Class: Constructor

```
class DoublyLinkedList {
private:
    Node* head;
    Node* tail;

public:
    DoublyLinkedList() : head(nullptr), tail(nullptr) {}
```

The DoublyLinkedList Class: insertEnd()

```cpp
// Function to insert a new node at the end of the list
 void insertEnd(BankAccount acc) {
     Node* newNode = new Node(acc);
     if (head == nullptr) {
         head = newNode;
         tail = newNode;
     } else {
         tail->next = newNode;
         newNode->prev = tail;
         tail = newNode;
     }
 }
```

The DoublyLinkedList Class: remove()

Four removal situations if account number found.

```cpp
    // Function to remove a node from the list by account number
    void remove(string accNum) {
        Node* current = head;
        while (current != nullptr) {
            if (current->account.getAccountNumber() == accNum) {
                if (current == head && current == tail) { // Only one node in the list
                    head = nullptr;
                    tail = nullptr;
                } else if (current == head) { // Removing the head node
                    head = current->next;
                    head->prev = nullptr;
                } else if (current == tail) { // Removing the tail node
                    tail = current->prev;
                    tail->next = nullptr;
                } else { // Removing a node from between
                    current->prev->next = current->next;
                    current->next->prev = current->prev;
                }
                delete current;
                return;
            }
            current = current->next;
        }
        cout << "Account not found." << endl;}
```

The DoublyLinkedList Class: remove()

```cpp
    // Function to remove a node from the list by account number
    void remove(string accNum) {
        Node* current = head;
        while (current != nullptr) {
            if (current->account.getAccountNumber() == accNum) {
                if (current == head && current == tail) { // Only one node in the list
                    head = nullptr;
                    tail = nullptr;
                } else if (current == head) { // Removing the head node
                    head = current->next;
                    head->prev = nullptr;
                } else if (current == tail) { // Removing the tail node
                    tail = current->prev;
                    tail->next = nullptr;
                } else { // Removing a node from between
                    current->prev->next = current->next;
                    current->next->prev = current->prev;
                }
                delete current;
                return;
            }
            current = current->next;
        }
        cout << "Account not found." << endl;}
```

The DoublyLinkedList Class: remove()

```
    // Function to remove a node from the list by account number
    void remove(string accNum) {
        Node* current = head;
        while (current != nullptr) {
            if (current->account.getAccountNumber() == accNum) {
                if (current == head && current == tail) { // Only one node in the list
                    head = nullptr;
                    tail = nullptr;
                } else if (current == head) { // Removing the head node
                    head = current->next;
                    head->prev = nullptr;
                } else if (current == tail) { // Removing the tail node
                    tail = current->prev;
                    tail->next = nullptr;
                } else { // Removing a node from between
                    current->prev->next = current->next;
                    current->next->prev = current->prev;
                }
                delete current;
                return;
            }
            current = current->next;
        }
        cout << "Account not found." << endl;}
```

The DoublyLinkedList Class: remove()

```cpp
    // Function to remove a node from the list by account number
    void remove(string accNum) {
        Node* current = head;
        while (current != nullptr) {
            if (current->account.getAccountNumber() == accNum) {
                if (current == head && current == tail) { // Only one node in the list
                    head = nullptr;
                    tail = nullptr;
                } else if (current == head) { // Removing the head node
                    head = current->next;
                    head->prev = nullptr;
                } else if (current == tail) { // Removing the tail node
                    tail = current->prev;
                    tail->next = nullptr;
                } else { // Removing a node from between
                    current->prev->next = current->next;
                    current->next->prev = current->prev;
                }
                delete current;
                return;
            }
            current = current->next;
        }
        cout << "Account not found." << endl;}
```

The DoublyLinkedList Class: remove()

```
// Function to remove a node from the list by account number
void remove(string accNum) {
    Node* current = head;
    while (current != nullptr) {
        if (current->account.getAccountNumber() == accNum) {
            if (current == head && current == tail) { // Only one node in the list
                head = nullptr;
                tail = nullptr;
            } else if (current == head) { // Removing the head node
                head = current->next;
                head->prev = nullptr;
            } else if (current == tail) { // Removing the tail node
                tail = current->prev;
                tail->next = nullptr;
            } else { // Removing a node from between
                current->prev->next = current->next;
                current->next->prev = current->prev;
            }
            delete current;
            return;
        }
        current = current->next;
    }
    cout << "Account not found." << endl;}
```

The DoublyLinkedList Class: withdrawBalance()

```
// Function to withdraw an amount from the balance of a bank account
    void withdrawBalance(string accNum, double amount) {
        Node* current = head;
        while (current != nullptr) {
            if (current->account.getAccountNumber() == accNum) {
                current->account.withdraw(amount);
                return;
            }
            current = current->next;
        }
        cout << "Account not found." << endl;
    }
```

The DoublyLinkedList Class: depositBalance()

```
    // Function to deposit an amount to the balance of a bank account
    void depositBalance(string accNum, double amount) {
        Node* current = head;
        while (current != nullptr) {
            if (current->account.getAccountNumber() == accNum) {
                current->account.deposit(amount);
                return;
            }
            current = current->next;
        }
        cout << "Account not found." << endl;
    }
```

The DoublyLinkedList Class: display()

```cpp
    // Function to display the elements of the list
    void display() {
        Node* temp = head;
        while (temp != nullptr) {
            cout << "Account Number: " << temp->account.getAccountNumber() << ",
                      Balance: " << temp->account.getBalance() << endl;
            temp = temp->next;
        }
        cout << endl;
    }
```

main() function

```
int main() {
    DoublyLinkedList dll;
    // Creating some bank accounts
    BankAccount acc1("12345", 1000.0);
    BankAccount acc2("67890", 2000.0);
    BankAccount acc3("54321", 3000.0);

    // Inserting bank accounts into the doubly linked list
    dll.insertEnd(acc1);
    dll.insertEnd(acc2);
    dll.insertEnd(acc3);

    // Displaying the initial list of bank accounts
    cout << "Initial List of Bank Accounts:" << endl;
    dll.display();

    // Removing an account and updating another account's balance
    dll.remove("67890");
    dll.updateBalance("54321", 4000.0);

    // Displaying the updated list of bank accounts
    cout << "Updated List of Bank Accounts:" << endl;
    dll.display();

    return 0;}
```