Федеральное государственное автономное образовательное учреждение высшего образования «Российский университет дружбы народов имени Патриса Лумумбы»

Факультет физико-математических и естественных наук

ЛАБОРАТОРНАЯ РАБОТА 1

по предмету «Технология программирования»

Студент: Чали Мутале
Ст.билет №: 1032239667
Группа: НКАбд-05–24

Москва, 2024 г.

Реализация списка на языке C++

1. Введение

Целью данной лабораторной работы является разработка программы на языке C++, которая реализует односвязный список на основе пользовательской структуры. Программа должна поддерживать основные операции по работе со списком, такие как добавление элементов в начало, конец, до и после указанного элемента, удаление элемента по имени, а также вывод содержимого списка на экран.

2. Описание структуры

Для реализации списка была создана структура Node, которая содержит следующие поля:

- name (тип std::string) — имя элемента;
- value (тип int) — целочисленное значение;
- data (тип double) — вещественное значение;
- next (тип Node*) — указатель на следующий элемент списка.

```
// Define the structure for a node in the linked list
struct Node {
    std::string name;  // Field to store a string
    int value;         // Field to store an integer
    double number;     // Field to store a double
    Node* next;        // Pointer to the next node in the list
};
```

3. Описание функций

i) Добавление элемента в начало списка:
Функция addToBeginning добавляет новый элемент в начало списка. Она принимает указатель на голову списка и значения для нового элемента.

```
// Function to add a node to the beginning of the list
void addToBeginning(Node*& head, std::string name, int value, double number) {
    Node* newNode = new Node;  // Create a new node
    newNode->name = name;      // Set the name field
    newNode->value = value;    // Set the value field
    newNode->number = number;  // Set the number field
    newNode->next = head;      // Point the new node to the current head
    head = newNode;            // Update the head to the new node
}
```

ii) Добавление элемента в конец списка:
Функция addToEnd добавляет новый элемент в конец списка. Если список пуст, новый элемент становится головой списка.

```
// Function to add a node to the end of the list
void addToEnd(Node*& head, std::string name, int value, double number) {
    Node* newNode = new Node;  // Create a new node
    newNode->name = name;      // Set the name field
    newNode->value = value;    // Set the value field
    newNode->number = number;  // Set the number field
    newNode->next = nullptr;   // Since it's the last node, next is nullptr

    if (head == nullptr) {     // If the list is empty
        head = newNode;        // The new node becomes the head
    } else {
        Node* temp = head;     // Start from the head
        while (temp->next != nullptr) {  // Traverse to the end of the list
            temp = temp->next;
        }
        temp->next = newNode;  // Link the last node to the new node
    }
}
```

iii) Добавление элемента после указанного элемента:
Функция addAfter добавляет новый элемент после элемента с указанным именем. Если элемент с таким именем не найден, новый элемент не добавляется.

```
// Function to add a node after a specific node in the list
void addAfter(Node* head, std::string afterName, std::string name, int value, double number) {
    Node* temp = head;  // Start from the head
    while (temp != nullptr && temp->name != afterName) {  // Find the node with the given name
        temp = temp->next;
    }
    if (temp != nullptr) {  // If the node is found
        Node* newNode = new Node;  // Create a new node
        newNode->name = name;      // Set the name field
        newNode->value = value;    // Set the value field
        newNode->number = number;  // Set the number field
        newNode->next = temp->next;  // Point the new node to the next node
        temp->next = newNode;      // Link the current node to the new node
    }
}
```

iv) Добавление элемента перед указанным элементом:
Функция addBefore добавляет новый элемент перед элементом с указанным именем. Если элемент с таким именем не найден, новый элемент не добавляется.

```
// Function to add a node before a specific node in the list
void addBefore(Node*& head, std::string beforeName, std::string name, int value, double number) {
    if (head == nullptr) return;  // If the list is empty, do nothing

    if (head->name == beforeName) {  // If the node to add before is the head
        addToBeginning(head, name, value, number);  // Add to the beginning
        return;
    }

    Node* temp = head;  // Start from the head
    while (temp->next != nullptr && temp->next->name != beforeName) {  // Find the node before the given name
        temp = temp->next;
    }
    if (temp->next != nullptr) {  // If the node is found
        Node* newNode = new Node;  // Create a new node
        newNode->name = name;      // Set the name field
        newNode->value = value;    // Set the value field
        newNode->number = number;  // Set the number field
        newNode->next = temp->next;  // Point the new node to the next node
        temp->next = newNode;      // Link the current node to the new node
    }
}
```

v) Удаление элемента по имени:
Функция deleteByName удаляет элемент с указанным именем из списка. Если элемент с таким именем не найден, список остается без изменений.

```
// Function to remove a node by its name
void removeByName(Node*& head, std::string name) {
    if (head == nullptr) return;  // If the list is empty, do nothing

    if (head->name == name) {  // If the node to remove is the head
        Node* temp = head;     // Store the head in a temporary variable
        head = head->next;     // Update the head to the next node
        delete temp;           // Delete the old head
        return;
    }

    Node* temp = head;  // Start from the head
    while (temp->next != nullptr && temp->next->name != name) {  // Find the node before the one to remove
        temp = temp->next;
    }
    if (temp->next != nullptr) {  // If the node is found
        Node* nodeToDelete = temp->next;  // Store the node to delete
        temp->next = temp->next->next;    // Link the previous node to the next node
        delete nodeToDelete;              // Delete the node
    }
}
```

vi) Вывод списка на экран
Функция displayList выводит содержимое списка на экран, отображая имя, целочисленное и вещественное значение каждого элемента.

```
// Function to print the contents of the list
void printList(Node* head) {
    Node* temp = head;  // Start from the head
    while (temp != nullptr) {  // Traverse the list
        std::cout << "Name: " << temp->name << ", Value: " << temp->value << ", Number: " << temp->number << std::endl;
        temp = temp->next;  // Move to the next node
    }
}
```

4. Основная программа

В функции main демонстрируется использование всех разработанных функций. Создается список, добавляются элементы, выполняется удаление элемента и вывод списка на экран.

```cpp
// Main function to demonstrate the list operations
int main() {
    Node* head = nullptr;  // Initialize the head of the list to nullptr

    // Add nodes to the list
    addToEnd(head, "Sasha", 10, 3.14);
    addToBeginning(head, "Lyosha", 20, 6.28);
    addAfter(head, "Sasha", "Katya", 30, 9.42);
    addBefore(head, "Katya", "Nikolai", 40, 12.56);

    // Print the initial list
    std::cout << "Initial List:" << std::endl;
    printList(head);

    // Remove a node by name and print the updated list
    removeByName(head, "Sasha");
    std::cout << "List after removing Sasha:" << std::endl;
    printList(head);

    return 0;
}
```

5. Результаты выполнения программы

При запуске программы вывод будет следующим:

```
cmutale@vbox:~/cpp$ g++ lab1.cpp -o lab1
cmutale@vbox:~/cpp$ ./lab1
Initial List:
Name: Lyosha, Value: 20, Number: 6.28
Name: Sasha, Value: 10, Number: 3.14
Name: Nikolai, Value: 40, Number: 12.56
Name: Katya, Value: 30, Number: 9.42
List after removing Sasha:
Name: Lyosha, Value: 20, Number: 6.28
Name: Nikolai, Value: 40, Number: 12.56
Name: Katya, Value: 30, Number: 9.42
cmutale@vbox:~/cpp$
```

6. Заключение

В ходе выполнения лабораторной работы была разработана программа на языке C++, реализующая односвязный список с поддержкой основных операций. Программа успешно прошла тестирование и продемонстрировала корректную работу всех функций.