МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания № 6

Тема:

«Двунаправленные динамические списки»

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Боргачев Т.М.

Группа: ИНБО-10-23

Москва – 2024

СОДЕРЖАНИЕ

Цель: получение знаний и практических навыков управления двунаправленным списком в программах на языке C++.

1 ФОРМУЛИРОВКА ЗАДАЧИ

1.1 Задание

Разработать многомодульную программу, которая демонстрирует выполнение всех операций, определенных вариантом 5, над линейным двунаправленным динамическим списком.

Требования к разработке:

1. Разработать структуру узла списка, структура информационной части узла определена вариантом. Для определения структуры узла списка, используйте тип struct или class. Сохраните определение структуры узла и прототипы функций в заголовочном файле.
2. Разработайте функции для выполнения операции над линейным двунаправленным динамическим списком:
   - создание списка;
   - вставку узла;
   - удаление узла;
   - вывод списка в двух направлениях (слева направо и справа налево);
   - поиск узла с заданным значением (операция должна возвращать указатель на узел с заданным значением).
3. Дополнительные операции над списком, указанные вариантом, оформите в виде функций и включите в отдельный файл с расширением cpp. Подключите к этому файлу заголовочный файл с определением структуры узла.
4. Разработайте программу, управляемую текстовым меню, и включите в меню демонстрацию выполнения всех операций задания и варианта.
5. Проведите тестирование операций.
   - Оцените сложность алгоритма первой дополнительной операции.

Вариант 5:

Тип информационной части:

- Номер счета в банке (20-значное число), дата, вид операции (приход или расход), сумма вклада.

Дополнительные операции:

- Вставка нового узла перед первым узлом.
- Удаление сведений по счету (всех узлов), у которого общая сумма вклада равна нулю (сумма по приходу, минус сумма по расходу).
- Создать новый список из исходного, которого будет содержать остаток по всем видам операций одного счета, указав вид операции – приход, и текущую дату.

2 ПРОЕКТИРОВАНИЕ И РЕАЛИЗАЦИЯ

2.1 Задание – преобразование списка

2.1.1 Структура узла

В соответствии с вариантом, каждый узел представляет собой банковскую ячейку с номером счета, датой операции, типом операции, полученными или изъятыми средствами. Тогда структура узла примет вид в соответствии с рис. 1.

```
struct Info {
    string accountNumber;
    string date;
    string operationType;
    double amount = 0;
};
```

Рисунок 1 – Структура узла

2.1.2 Базовые операции

Блок-схемы базовых операций для списка: проверка на пустоту, добавление элемента, поиск элемента, вывод слева направо, удаление элемента, вывод справа налево – соответственно представлены на рис. 2, 3, 4, 5, 6 и 7.

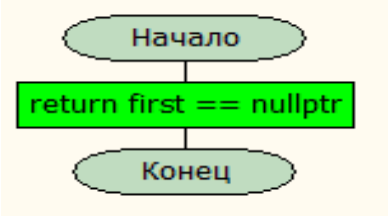

Рисунок 2 – Метод проверки на пустоту

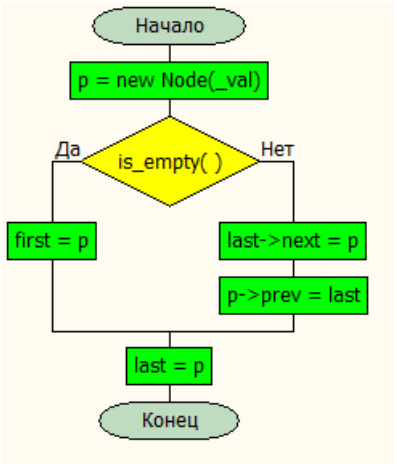

Рисунок 3 – Метод добавления элемента в конец списка

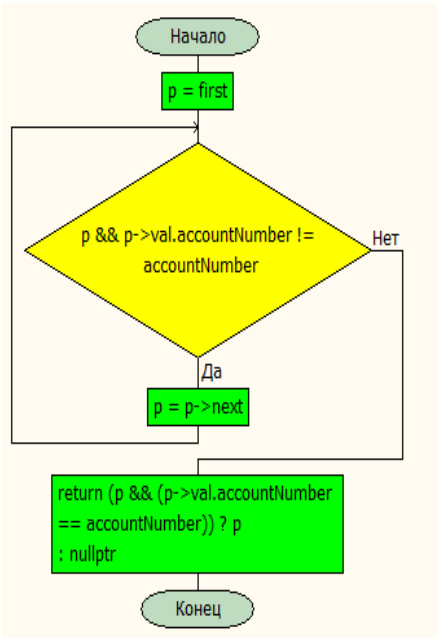

Рисунок 4 – Метод поиска объекта в списке

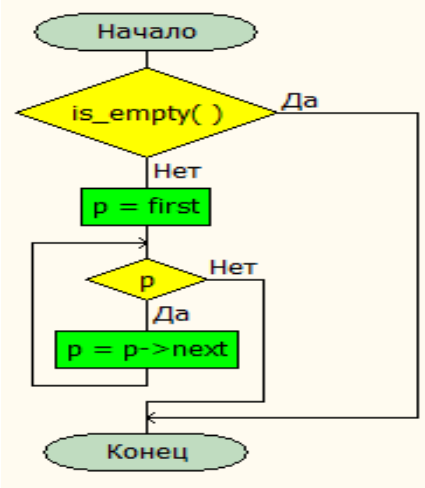

Рисунок 5 – Метод вывода списка слева направо

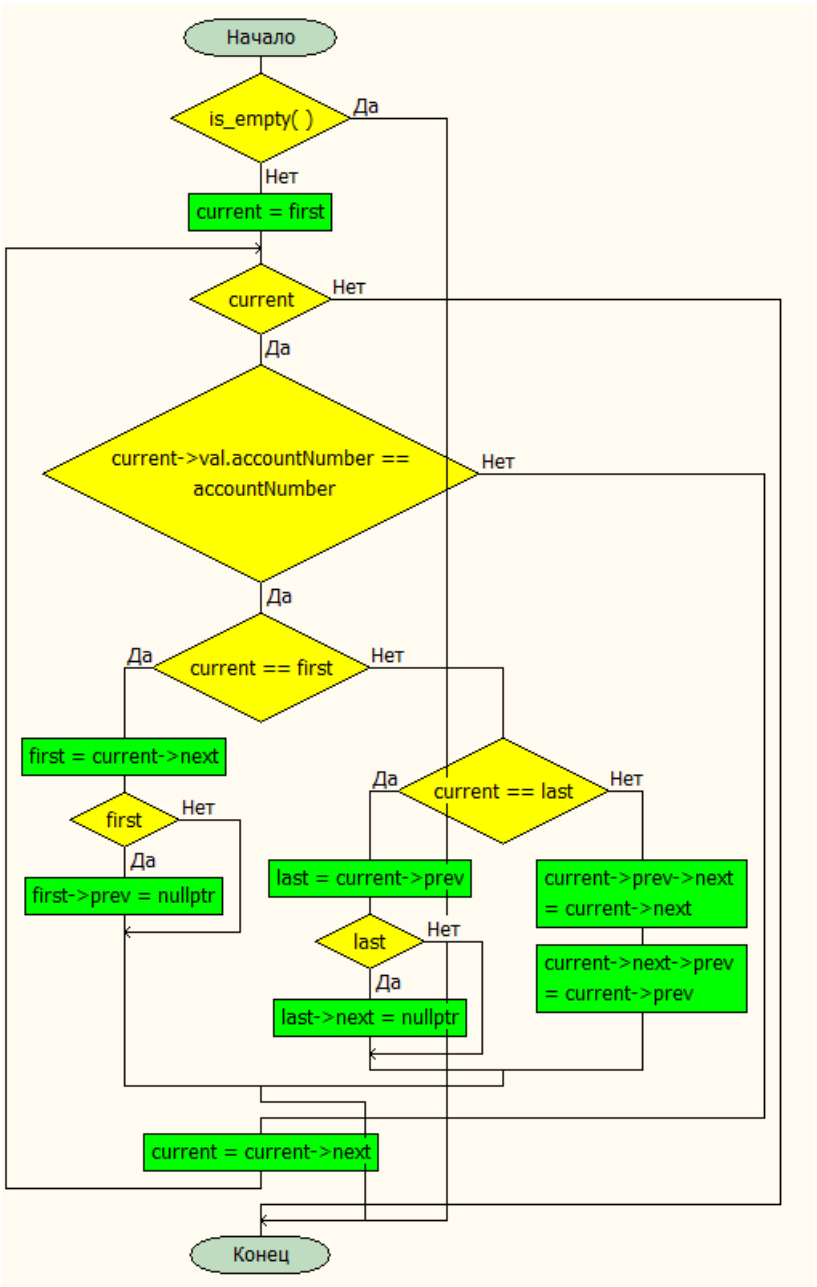

Рисунок 6 – Метод удаления объекта из списка

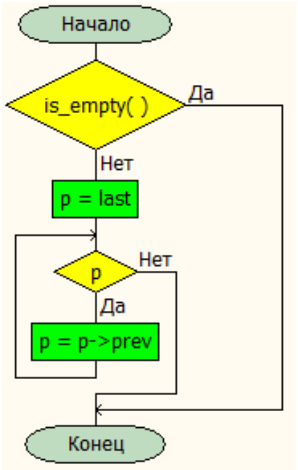

Рисунок 7 – Метод вывода списка справа налево

2.1.3 Структура списка

Двунаправленный список состоит из значения узла, указателя на следующий объект и указателя на предыдущий объект. Тогда структура списка будет выглядеть как представлено на рис. 8.

```
struct Node {
    Info val;
    Node* next;
    Node* prev;
    Node(Info _val) : val(_val), next(nullptr), prev(nullptr) {}
};
```

Рисунок 8 – Структура списка

2.1.4 Описание дополнительных методов для списка

Блок-схемы дополнительных методов для списка представлены на рис. 9, 10 и 11 соответственно.

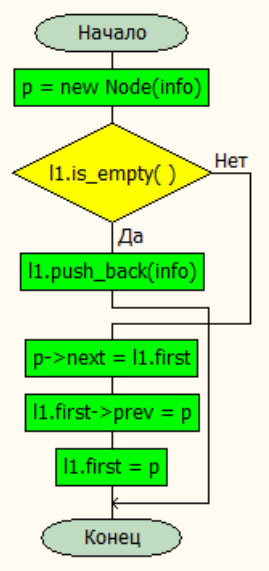

Рисунок 9 – Метод вставки объекта в начало списка

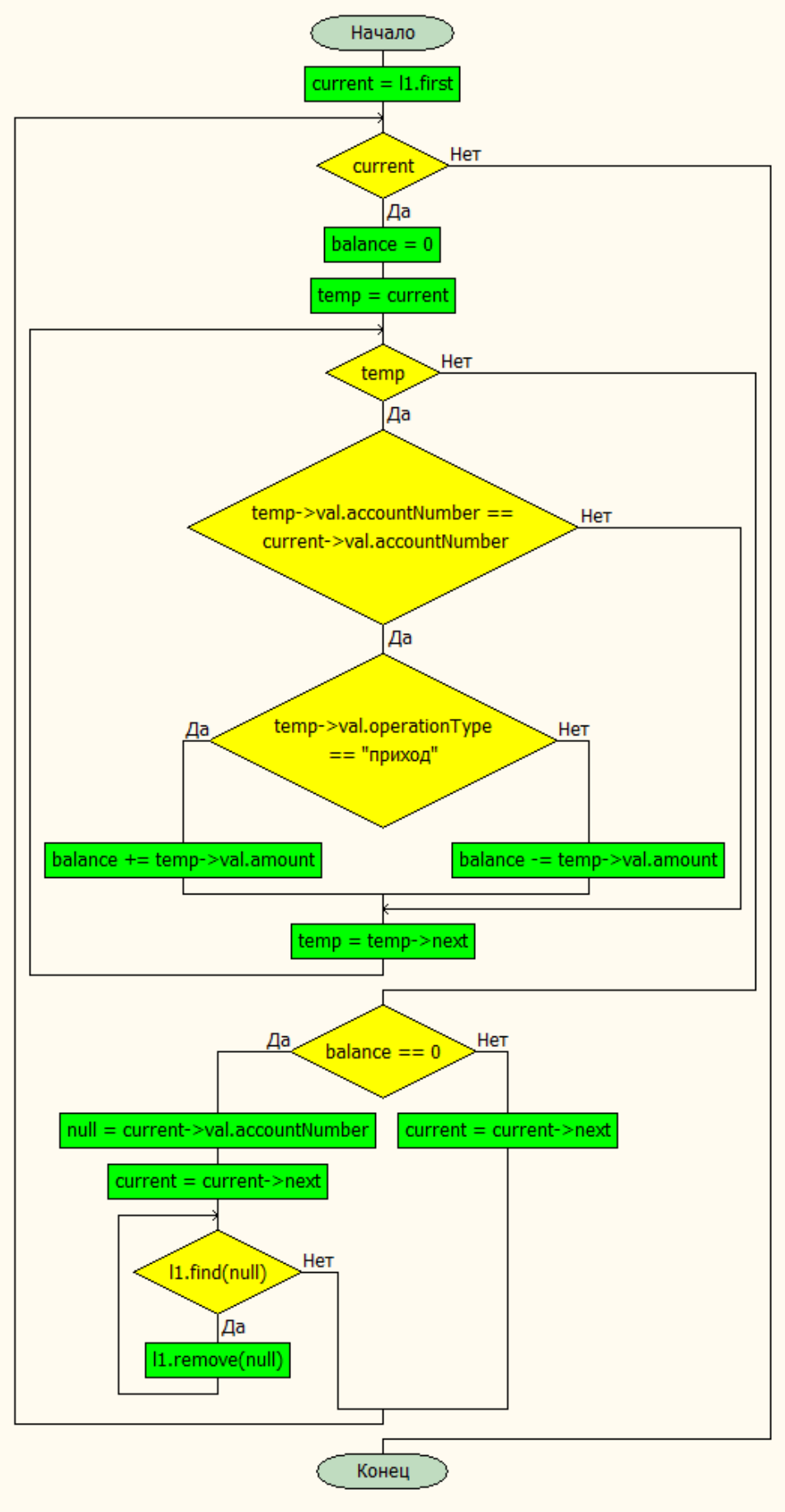

Рисунок 10 – Метод удаления счетов с нулевым балансом

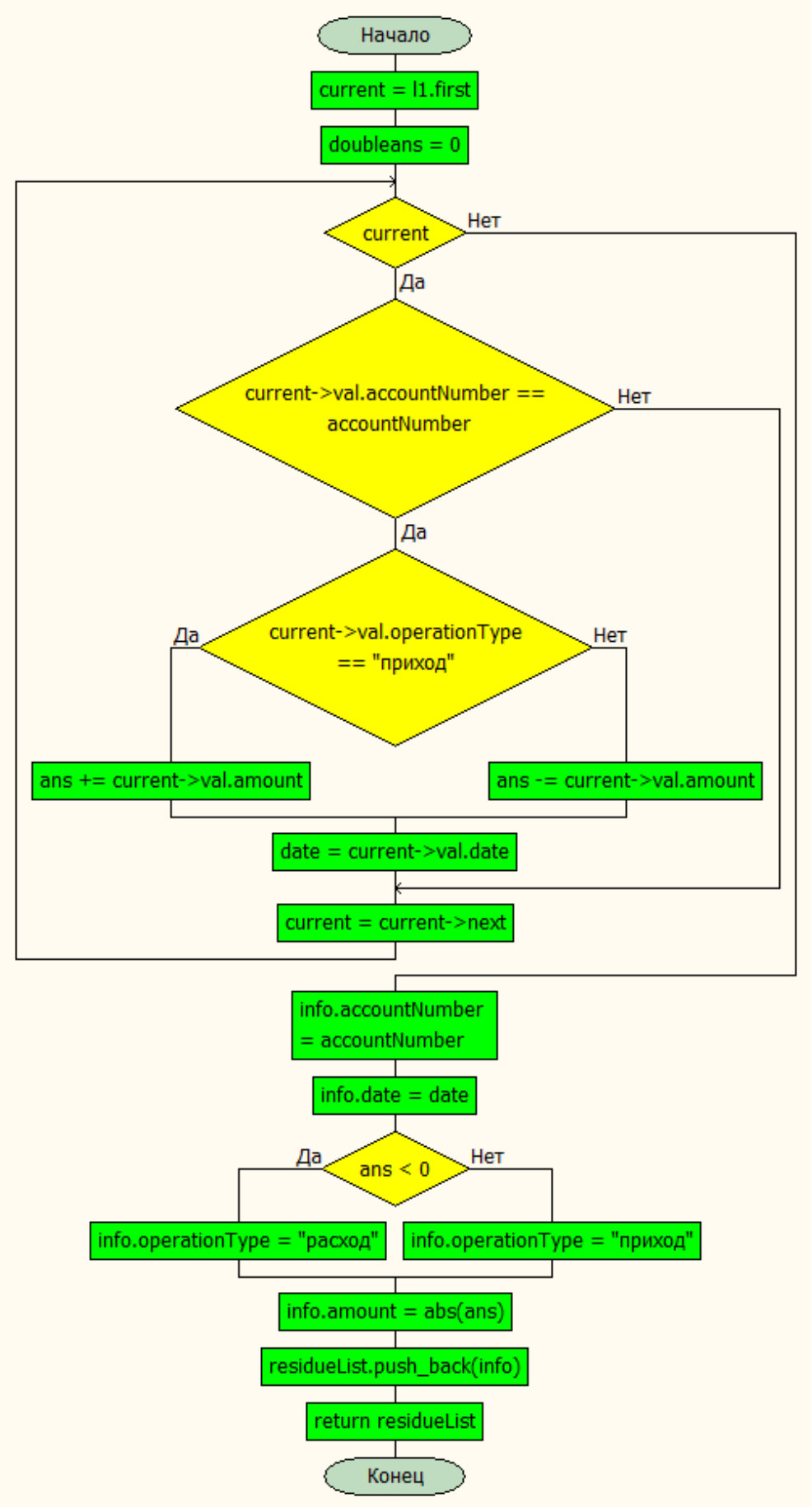

Рисунок 11 – Метод создания списка конечного состояния счета

2.1.5 Ожидаемые тесты программы

Алгоритм программы построим так: создадим два пустых списка l1 и l2, l2 оставим пустым, в l1 запишем некоторое количество банковских счетов.

Начальные данные для l1 представлены на рис. 12.

```
10413857354913453258, 24.03.2024, приход, 0
10413851254913453258, 23.03.2024, расход, 10000
10413147354913453258, 22.03.2024, приход, 0
20413147354913453258, 22.03.2024, приход, 20
```

Рисунок 12 – Начальные данные

Далее проверим оба списка на пустоту и будем вызывать все методы для l1 последовательно. Ожидаемый результат проверки l2 на пустоту – правда. Таблица ожидаемых результатов для l1 представлена в табл. 1.

Таблица 1 – Ожидаемые результаты

Метод	Результат для l1
is_empty()	true
push_back(Info _val)	10413857354913453258, 24.03.2024, приход, 0 10413851254913453258, 23.03.2024, расход, 10000 10413147354913453258, 22.03.2024, приход, 0 20413147354913453258, 22.03.2024, приход, 20
is_empty()	false
find(“10413851254913453258”)	Указатель на 3ий узел
remove(“20413147354913453258”)	10413857354913453258, 24.03.2024, приход, 0 10413851254913453258, 23.03.2024, расход, 10000 10413147354913453258, 22.03.2024, приход, 0
left_print()	Account: 104138573549134 5325, Date: 24.03.2024, Operation: приход, Amount: 0 Account: 104138512549134 5325, Date: 23.03.2024, Operation: расход, Amount: 10000 Account: 104131473549134 5325, Date: 22.03.2024, Operation: приход, Amount: 0

Метод	Результат для l1
right_print()	Account: 1041314735491345325, Date: 22.03.2024, Operation: приход, Amount: 0 Account: 1041385125491345325, Date: 23.03.2024, Operation: расход, Amount: 10000 Account: 1041385735491345325, Date: 24.03.2024, Operation: приход, Amount: 0
Insert("11111111111111111111, 26.03.2024, расход, 0",&l1)	11111111111111111111, 26.03.2024, расход, 0 10413857354913453258, 24.03.2024, приход, 0 10413851254913453258, 23.03.2024, расход, 10000 10413147354913453258, 22.03.2024, приход, 0
removeZeroBalanceAccounts(l1)	10413851254913453258, 23.03.2024, расход, 10000
createResidueList("1041385125491345325", l1)	Account: 1041385125491345325, Date: 23.03.2024, Operation: расход, Amount: 10000

2.2 Реализация алгоритма на языке C++

Реализация основных методов для списка представлена на рис. 13 и 14.

```
void DoublyLinkedList::left_print() {
    if (is_empty()) { return; }
    Node* p = first;
    while (p) {
        std::cout << "Account: " << p->val.accountNumber << ", Date: " << p->val.date << ", Operation: " << p->val.operationType << ", Amount: " << p->val.amount << std::endl;
        p = p->next;
    }
}
void DoublyLinkedList::right_print() {
    if (is_empty()) { return; }
    Node* p = last;
    while (p) {
        std::cout << "Account: " << p->val.accountNumber << ", Date: " << p->val.date << ", Operation: " << p->val.operationType << ", Amount: " << p->val.amount << std::endl;
        p = p->prev;
    }
}
```

Рисунок 13 – Реализация стандартных методов

```cpp
void DoublyLinkedList::push_back(Info _val) {
    Node* p = new Node(_val);
    if (is_empty()) {
        first = p;
    }
    else {
        last->next = p;
        p->prev = last;
    }
    last = p;
}
Node* DoublyLinkedList::find(std::string accountNumber) {
    Node* p = first;
    while (p && p->val.accountNumber != accountNumber) {
        p = p->next;
    }
    return (p && (p->val.accountNumber == accountNumber)) ? p : nullptr;
}
void DoublyLinkedList::remove(std::string accountNumber) {
    if (is_empty()) { return; }
    Node* current = first;
    while (current) {
        if (current->val.accountNumber == accountNumber) {
            if (current == first) {
                first = current->next;
                if (first)
                    first->prev = nullptr;
            }
            else if (current == last) {
                last = current->prev;
                if (last)
                    last->next = nullptr;
            }
            else {
                current->prev->next = current->next;
                current->next->prev = current->prev;
            }
            delete current;
            return;
        }
        current = current->next;
    }
    std::cout << "No element with account number " << accountNumber << std::endl;
}
```

Рисунок 14 – Реализация стандартных методов

Реализация дополнительных методов представлена на рис. 15 и 16.

```
void Insert(Info info, DoublyLinkedList& l1) {
    Node* p = new Node(info);
    if (l1.is_empty()) {
        l1.push_back(info);
        return;
    }
    p->next = l1.first;
    l1.first->prev = p;
    l1.first = p;
}
void removeZeroBalanceAccounts(DoublyLinkedList &l1) {
    Node* current = l1.first;
    while (current) {
        double balance = 0;
        Node* temp = current;
        while (temp) {
            if (temp->val.accountNumber == current->val.accountNumber) {
                if (temp->val.operationType == "приход") {
                    balance += temp->val.amount;
                }
                else {
                    balance -= temp->val.amount;
                }
            }
            temp = temp->next;
        }
        if (balance == 0) {
            string null = current->val.accountNumber;
            current = current->next;
            while(l1.find(null))l1.remove(null);
        }
        else {
            current = current->next;
        }
    }
}
```

Рисунок 15 – Реализация дополнительных методов

```
DoublyLinkedList createResidueList(const std::string& accountNumber, DoublyLinkedList& l1) {
    DoublyLinkedList residueList;
    Node* current = l1.first;
    string date;
    double ans=0;
    while (current) {
        if (current->val.accountNumber == accountNumber) {
            if (current->val.operationType == "приход") ans += current->val.amount;
            else ans -= current->val.amount;
            date = current->val.date;
        }
        current = current->next;
    }
    Info info;
    info.accountNumber = accountNumber;
    info.date = date;
    if (ans < 0) info.operationType = "расход";
    else info.operationType = "приход";
    info.amount = abs(ans);
    residueList.push_back(info);
    return residueList;
}
```

Рисунок 16 – Реализация дополнительных методов

Основной алгоритм программы для проверки корректности работы всех методов представлен на рис. 17.

```
int main() {
    setlocale(LC_ALL, "rus");
    // Создание пустого списка
    DoublyLinkedList l1;
    DoublyLinkedList l2;
    if (l2.is_empty()) cout << "Список l2 пуст" << endl;
    // Чтение данных из файла и запись их в список l1
    ifstream file("l1.txt");
    string line;
    while (getline(file, line)) {
        size_t ind1, ind2, ind3;
        ind1 = line.find(',', 0);
        ind2 = line.find(',', ind1 + 1);
        ind3 = line.find(',', ind2 + 1);
        // Преобразование строки в структуру OperationInfo
        Info info;
        info.accountNumber = line.substr(0, ind1 - 1);
        info.date = line.substr(ind1 + 2, ind2 - 2 - ind1);
        info.operationType = line.substr(ind2 + 2, ind3 - 2 - ind2);
        info.amount = stoi(line.substr(ind3 + 2));
        // Вставка информации в список
        Insert(info, l1);
    }
    file.close();
    if (l1.is_empty()) cout << "Список l1 пуст" << endl;
    else cout << "Список l1 не пуст" << endl;
    // Вывод содержимого списка до удаления счетов с нулевым балансом
    cout << "Содержимое списка до удаления счетов с нулевым балансом:" << endl;
    l1.left_print();
    cout << endl;
    l1.right_print();
    if (l1.find("10413851254913345325")) cout << "Узел найден\n";
    else cout << "Узел не найден\n";
    // Удаление счетов с нулевым балансом
    removeZeroBalanceAccounts(l1);

    // Вывод содержимого списка после удаления счетов с нулевым балансом
    cout << "Содержимое списка после удаления счетов с нулевым балансом:" << endl;
    l1.left_print();

    // Создание нового списка из исходного с остатком по всем видам операций одного счета
    string accountNumber="10413851254913345325";
    DoublyLinkedList residueList = createResidueList(accountNumber, l1);

    // Вывод содержимого нового списка с остатком по всем видам операций одного счета
    cout << "Содержимое нового списка с остатком по всем видам операций одного счета:" << endl;
    residueList.left_print();

    return 0;
}
```

Рисунок 17 – Основной алгоритм программы

2.3 Тестирование программы

Тестовые данные для программы представлены на рис. 18.

```
1041385735491345325 8, 24.03.2024, приход, 12000
1041385125491345325 8, 23.03.2024, расход, 10000
1041314735491345325 8, 22.03.2024, приход, 100
1041385735491345325 8, 24.03.2024, расход, 12000
1041385125491345325 8, 23.03.2024, расход, 0
1041314735491345325 8, 22.03.2024, приход, 100
1041385735491345325 8, 21.03.2024, приход, 5000
1041385735491345325 8, 20.03.2024, расход, 7000
1041385735491345325 8, 19.03.2024, приход, 0
1041314735491345325 8, 18.03.2024, приход, 200
1041385125491345325 8, 17.03.2024, расход, 1500
1041385735491345325 8, 16.03.2024, приход, 9000
1041314735491345325 8, 15.03.2024, приход, 0
1041385125491345325 8, 14.03.2024, расход, 2000
1041385735491345325 8, 13.03.2024, приход, 3000
1041314735491345325 8, 12.03.2024, приход, 1500
1041385125491345325 8, 11.03.2024, расход, 4000
1041385735491345325 8, 10.03.2024, расход, 24000
1041314735491345325 8, 09.03.2024, приход, 2500
1041385125491345325 8, 08.03.2024, расход, 6000
```

Рисунок 18 – Входные данные

Результат тестирования алгоритма на входных данных представлен на рис. 19 и 20.

```
Список l2 пуст
Список l1 не пуст
Содержимое списка до удаления счетов с нулевым балансом:
Слева направо
Account: 1041385125491345325, Date: 08.03.2024, Operation: расход, Amount: 6000
Account: 1041314735491345325, Date: 09.03.2024, Operation: приход, Amount: 2500
Account: 1041385735491345325, Date: 10.03.2024, Operation: расход, Amount: 24000
Account: 1041385125491345325, Date: 11.03.2024, Operation: расход, Amount: 4000
Account: 1041314735491345325, Date: 12.03.2024, Operation: приход, Amount: 1500
Account: 1041385735491345325, Date: 13.03.2024, Operation: приход, Amount: 3000
Account: 1041385125491345325, Date: 14.03.2024, Operation: расход, Amount: 2000
Account: 1041314735491345325, Date: 15.03.2024, Operation: приход, Amount: 0
Account: 1041385735491345325, Date: 16.03.2024, Operation: приход, Amount: 9000
Account: 1041385125491345325, Date: 17.03.2024, Operation: расход, Amount: 1500
Account: 1041314735491345325, Date: 18.03.2024, Operation: приход, Amount: 200
Account: 1041385735491345325, Date: 19.03.2024, Operation: приход, Amount: 0
Account: 1041385735491345325, Date: 20.03.2024, Operation: расход, Amount: 7000
Account: 1041385735491345325, Date: 21.03.2024, Operation: приход, Amount: 5000
Account: 1041314735491345325, Date: 22.03.2024, Operation: приход, Amount: 100
Account: 1041385125491345325, Date: 23.03.2024, Operation: расход, Amount: 0
Account: 1041385735491345325, Date: 24.03.2024, Operation: расход, Amount: 12000
Account: 1041314735491345325, Date: 22.03.2024, Operation: приход, Amount: 100
Account: 1041385125491345325, Date: 23.03.2024, Operation: расход, Amount: 10000
Account: 1041385735491345325, Date: 24.03.2024, Operation: приход, Amount: 12000

Справа налево
Account: 1041385735491345325, Date: 24.03.2024, Operation: приход, Amount: 12000
Account: 1041385125491345325, Date: 23.03.2024, Operation: расход, Amount: 10000
Account: 1041314735491345325, Date: 22.03.2024, Operation: приход, Amount: 100
Account: 1041385735491345325, Date: 24.03.2024, Operation: расход, Amount: 12000
Account: 1041385125491345325, Date: 23.03.2024, Operation: расход, Amount: 0
Account: 1041314735491345325, Date: 22.03.2024, Operation: приход, Amount: 100
Account: 1041385735491345325, Date: 21.03.2024, Operation: приход, Amount: 5000
Account: 1041385735491345325, Date: 20.03.2024, Operation: расход, Amount: 7000
Account: 1041385735491345325, Date: 19.03.2024, Operation: приход, Amount: 0
Account: 1041314735491345325, Date: 18.03.2024, Operation: приход, Amount: 200
Account: 1041385125491345325, Date: 17.03.2024, Operation: расход, Amount: 1500
Account: 1041385735491345325, Date: 16.03.2024, Operation: приход, Amount: 9000
Account: 1041314735491345325, Date: 15.03.2024, Operation: приход, Amount: 0
Account: 1041385125491345325, Date: 14.03.2024, Operation: расход, Amount: 2000
Account: 1041385735491345325, Date: 13.03.2024, Operation: приход, Amount: 3000
Account: 1041314735491345325, Date: 12.03.2024, Operation: приход, Amount: 1500
Account: 1041385125491345325, Date: 11.03.2024, Operation: расход, Amount: 4000
Account: 1041385735491345325, Date: 10.03.2024, Operation: расход, Amount: 24000
Account: 1041314735491345325, Date: 09.03.2024, Operation: приход, Amount: 2500
Account: 1041385125491345325, Date: 08.03.2024, Operation: расход, Amount: 6000
Узел найден
```

Рисунок 19 – Результат тестирования основных методов и метода Insert

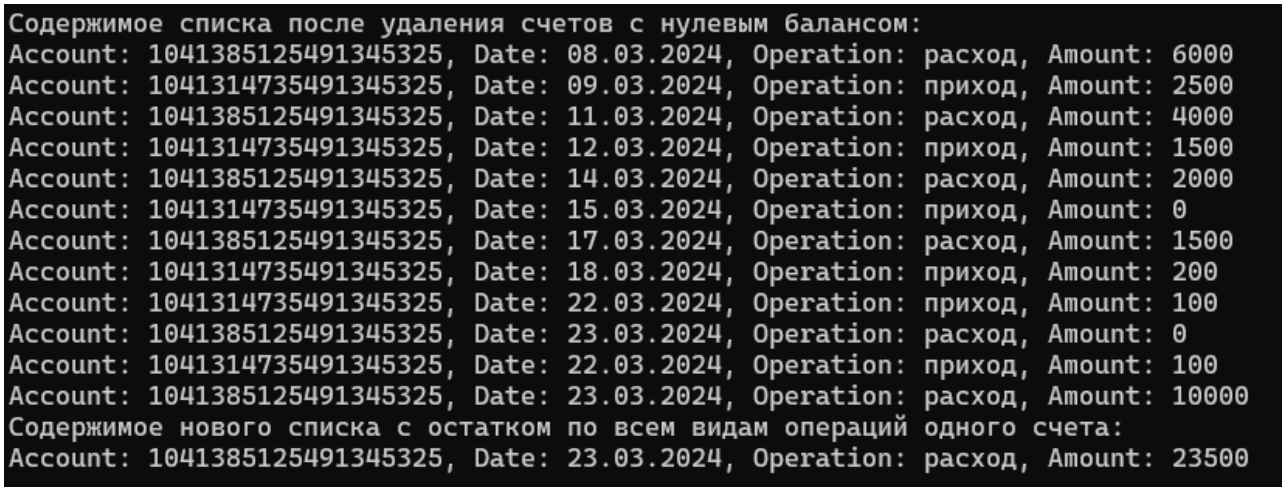

Рисунок 20 – Результат тестирования дополнительных методов и метода push_back

Таким образом, алгоритм обрабатывает входные данные корректно.

3 ВЫВОДЫ

В ходе работы были реализованы структуры для работы с двунаправленными списками, написаны основные методы для работы с данными в них, а также 3 дополнительных функции для обработки данных, а именно: вставка объекта в начало списка, удаление из списка узлов с нулевыми счетами, создание нового списка для конечного результата одного из счетов.

Таким образом, были получены теоретические знания о двунаправленных списках, а также практические навыки и умения для работы с ними, и их реализации в программах на C++.

4 ИНФОРМАЦИОННЫЕ ИСТОЧНИКИ

1. Сартаков М.В., ПР-1.1 (Теоретическая сложность алгоритма)М., МИРЭА — Российский технологический университет – 12 с. - URL: https://online-edu.mirea.ru/pluginfile.php?file=%2F1042738%2Fmod_assign%2Fintroattachment%2F0%2FПР1.1%20%28Теоретическая%20сложность%20алгоритма%29.pdf&forcedownload=1 (дата обращения: 15.02.2024). - Режим доступа: Электронно-облачная система – Cloud MIREA РТУ МИРЭА. - Текст: электронный.

2. Рысин М.Л., Сартаков М.В., Туманова М.Б., Введение в структуры и алгоритмы обработки данных. Ч. 1 - учебное пособие, 2022, МИРЭА – Российский технологический университет. – 2022, 109с. – URL: file:///C:/Users/borga/Downloads/Рысин%20М.Л.%20и%20др.%20Введение%20в%20структуры%20и%20алгоритмы%20обработки%20данных.%20Ч.%201%20-%20учебное%20пособие,%202022.pdf (дата обращения: 15.02.2024). – Режим доступа: Электронно-облачная система – Cloud MIREA РТУ МИРЭА. - Текст: электронный.