

МИНОБРНАУКИ РОССИИ

*Федеральное государственное бюджетное образовательное учреждение высшего образования*

***«МИРЭА – Российский технологический университет»***

**РТУ МИРЭА**



Отчет по выполнению практического задания № 6

**Тема:**

«Двунаправленный динамический список»

Дисциплина: «Структуры и алгоритмы обработки данных»

| Выполнил студент группы: ИКБО-73-23 |  |  | Полянский. М. Д |
| --- | --- | --- | --- |
| Принял ассистент |  |  | Золотухин. С. А |

Москва – 2024

Содержание

[**Тема:** 1](#_gjdgxs)

[**1.**](#_30j0zll) **Задание.** 3

[**1.**](#_1fob9te) **Разработать многомодульную программу,** которая демонстрирует выполнение всех операций, определенных вариантом, над линейным двунаправленным динамическим списком. 3

[**2. Требования к разработке:** 3](#_3znysh7)

[**3. Функции для выполнения операций над списком:** 4](#_2et92p0)

[**2.**](#_tyjcwt) **Структура узла в соответствии с вариантом.** 4

[2.1.](#_3dy6vkm) **Блок-схемы операций над линейным списком** 6

[2.2. Структура данных 11](#_1t3h5sf)

[2.3. Словесное описание алгоритмов. 12](#_4d34og8)

[1) insert\_node(Node\* cur, Node\* insert). 12](#_2s8eyo1)

[2) find\_value(Node\* head, string value). 12](#_17dp8vu)

[3) delete\_node(Node\* cur). 12](#_3rdcrjn)

[4) print\_node(Node\* node). 12](#_26in1rg)

[5) print\_list\_back(Node\* head). 12](#_lnxbz9)

[6) print\_list(Node\* head). 12](#_35nkun2)

[7) insert\_task(Node\* head). 12](#_1ksv4uv)

[8) remove\_nums(Node\* head, int remove). 13](#_44sinio)

[9) make\_new\_list(Node\* head1, Node\* head2) 13](#_2jxsxqh)

[2.4. Таблица тестов для каждой операции. 13](#_z337ya)

[**3.**](#_3j2qqm3) **Код программы на языке С++** 14

[**1.**](#_1y810tw) **Linkedlist.h** 14

[**2.**](#_4i7ojhp) **Linkedlist.cpp** 15

[**3.**](#_2xcytpi) **main.cpp** 19

[**4.**](#_1ci93xb) **Результаты тестирования** 21

[**5.**](#_3whwml4) **Вывод** 23

[**6.**](#_2bn6wsx) **Литература** 23

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

1. **Задание.**
2. **Разработать многомодульную программу,** которая демонстрирует выполнение всех операций, определенных вариантом, над линейным двунаправленным динамическим списком.

**2. Требования к разработке:**

- Программа должна быть многомодульной.

- Структура узла и операции над списком должны быть реализованы в отдельных заголовочных файлах и файлах с исходным кодом.

**3. Функции для выполнения операций над списком:**

- Создание списка: функция `createList()` должна инициализировать голову и хвост списка как `nullptr`, создавая пустой список.

- Вставка узла: функция `insertNode(int cashRegisterNumber)` должна создавать новый узел с указанным номером кассы и добавлять его в конец списка. Если список пустой, новый узел становится и головой, и хвостом списка.

- Удаление узла: функция `deleteNode(int cashRegisterNumber)` должна удалять узел с указанным номером кассы из списка. Если такой узел не найден, должно выводиться сообщение об ошибке.

- Вывод списка в двух направлениях: функции `printForward()` и `printBackward()` должны выводить содержимое списка, начиная с головы и двигаясь к хвосту, либо начиная с хвоста и двигаясь к голове соответственно. Для каждого узла отображается номер кассы и список проведенных операций.

- Поиск узла с заданным значением: функция `searchNode(int cashRegisterNumber)` должна находить узел с указанным номером кассы и возвращать указатель на него. Если узел не найден, должен возвращаться `nullptr`.

- Вставить новый узел перед первым узлом с таким же ключом: функция `insertBeforeFirstOccurrence(int cashRegisterNumber)` должна вставлять новый узел перед первым узлом с указанным номером кассы. Если такого узла нет, новый узел должен быть вставлен перед первым узлом с номером кассы больше указанного.

- Удалить узлы с указанным номером кассы: функция `deleteNodesByCashRegisterNumber(int cashRegisterNumber)` должна удалять все узлы с указанным номером кассы из списка.

1. **Структура узла в соответствии с вариантом.**

Узел содержит следующие поля:

Поля целочисленного типа:

- cashRegisterNumber: номер кассы в магазине.

Поля типа указателя на узел:

- operations: указатель на список проведенных кассой операций.

- prev: указатель на предыдущий узел списка.

- next: указатель на следующий узел списка.

Структура Operation, хранящая информацию о проведенной операции, содержит следующие поля:

Поля строкового типа:

- productCode: код товара.

- unit: единица измерения (например, кг или штуки).

- date: дата операции.

Поля числового типа:

- quantity: количество товара.

- pricePerUnit: цена за единицу товара.

- totalAmount: сумма к оплате за товар.

Поле типа указателя на следующую операцию:

- nextOperation: указатель на следующую операцию в списке проведенных операций кассы.

Таким образом, структура узла включает информацию о кассе в магазине и список проведенных кассой операций, а каждая операция хранится в структуре Operation.

* 1. **Блок-схемы операций над линейным списком**

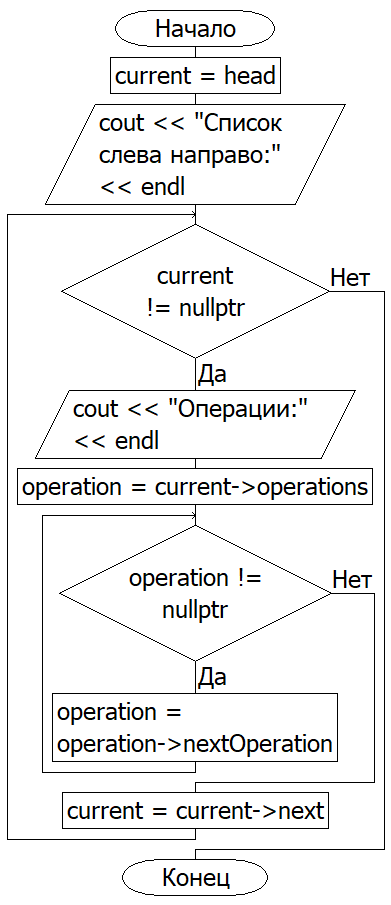


Рисунок 1 – Блок-схема функции вывода списка.

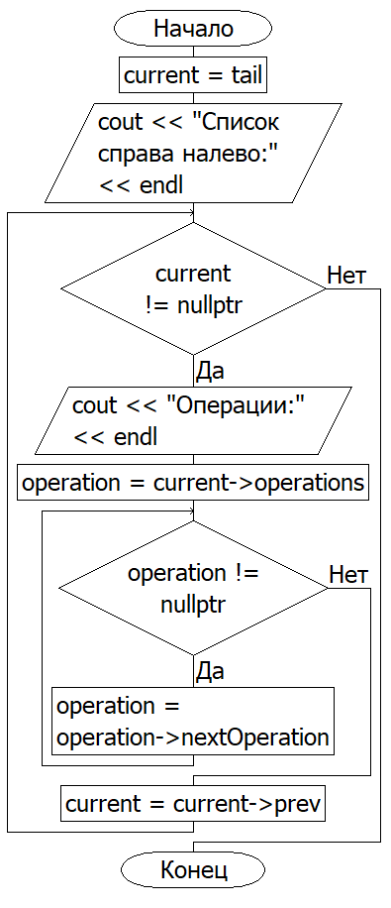


Рисунок 2 – Блок-схема функции вывода списка в обратном порядке.

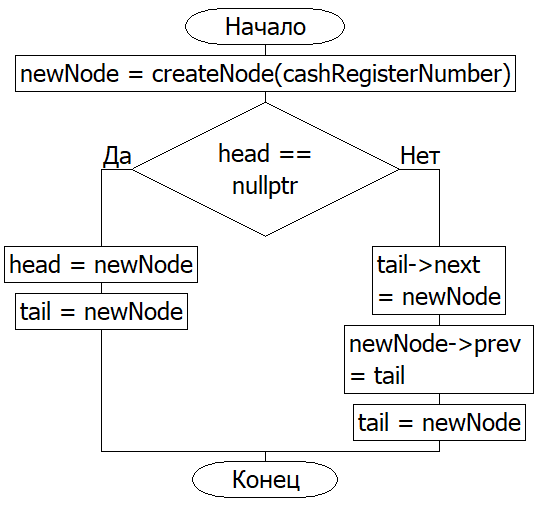


Рисунок 3 – Блок-схема функции вставки нового узла в список, в соответствии с условием.

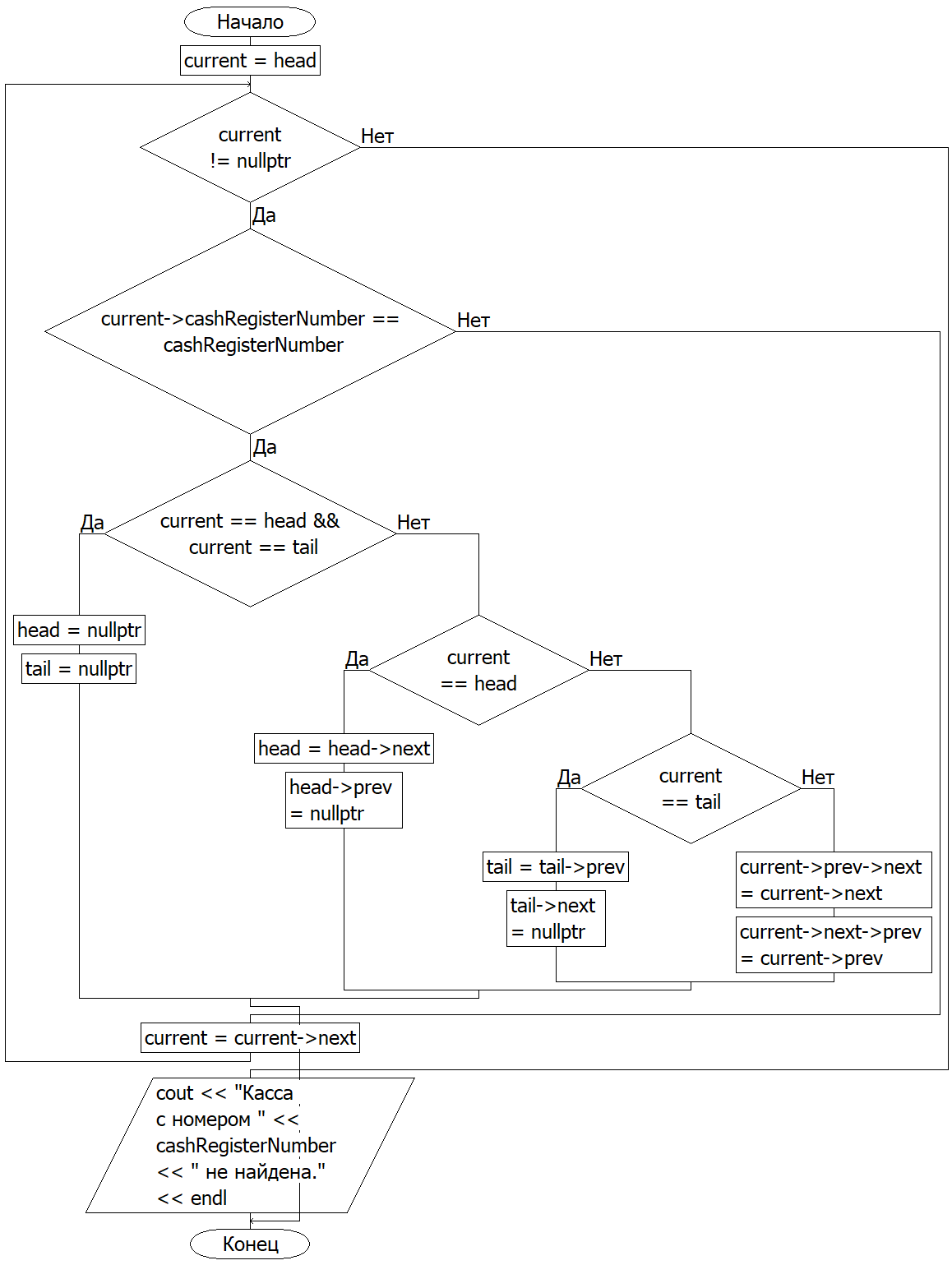


Рисунок 4 – Блок-схема функции удаления узлов с указанным номером группы.

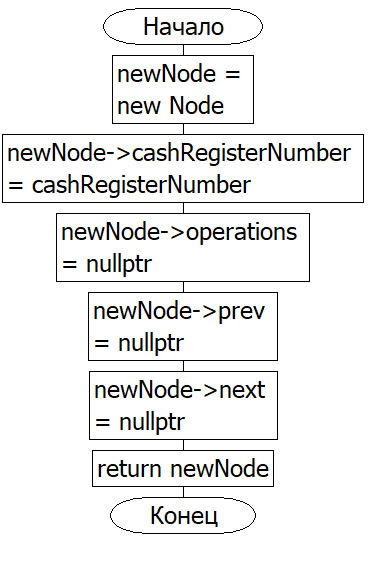


Рисунок 5 – Блок-схема функции поиск узла в списке по указателю.

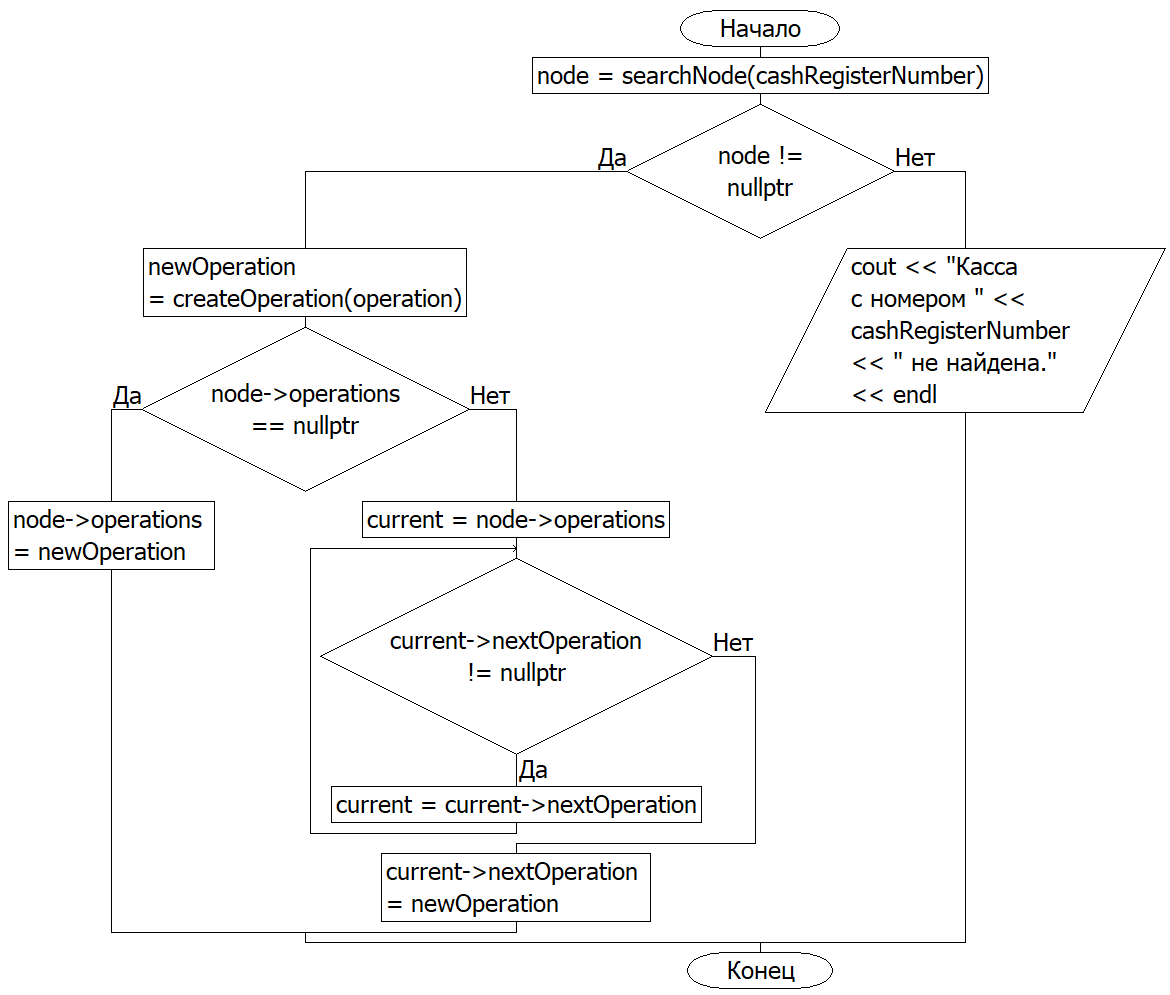


Рисунок 6 – Блок-схема функции добавление операции к указанной кассе.

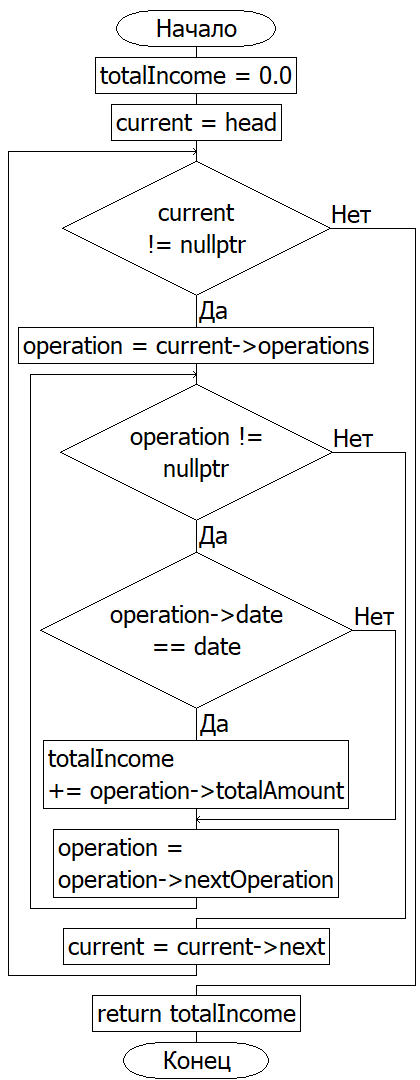


Рисунок 7 – Блок-схема функции определение прихода по кассе в указанную дату.

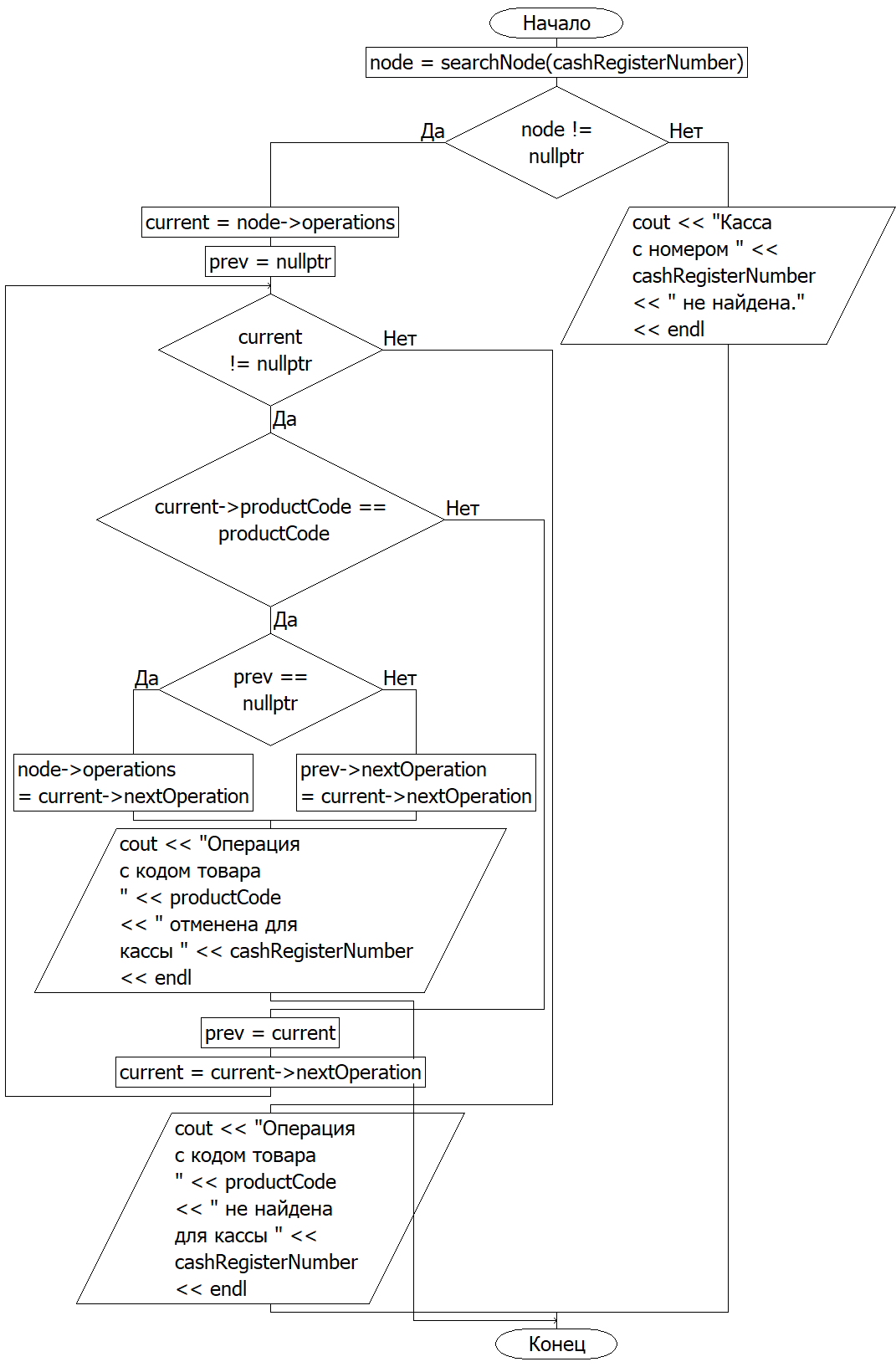


Рисунок 8 – Блок-схема функции отмена операции по указанной кассе и коду товара.

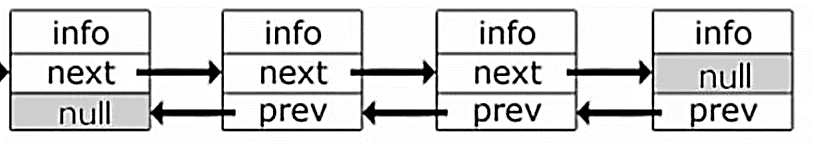
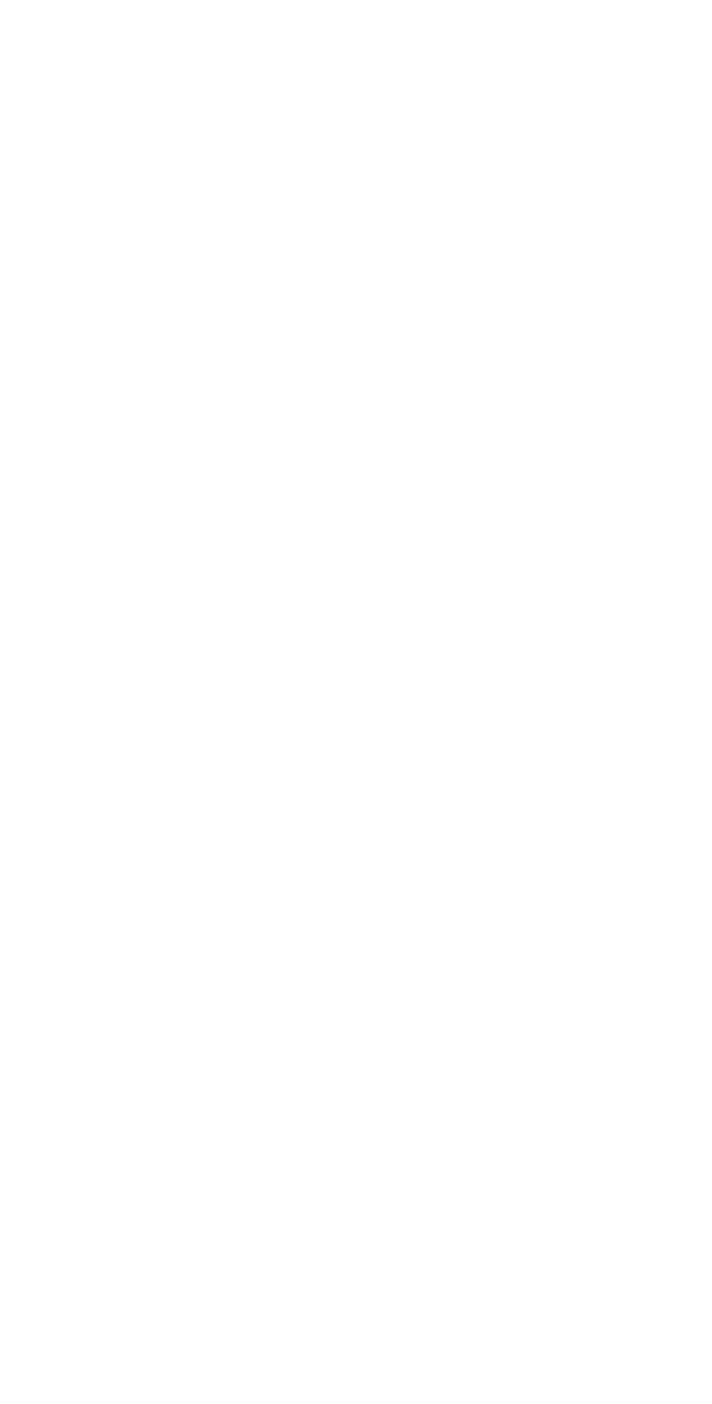
* 1. Структура данных

Рисунок 9 – структура данных.

* 1. Словесное описание алгоритмов.

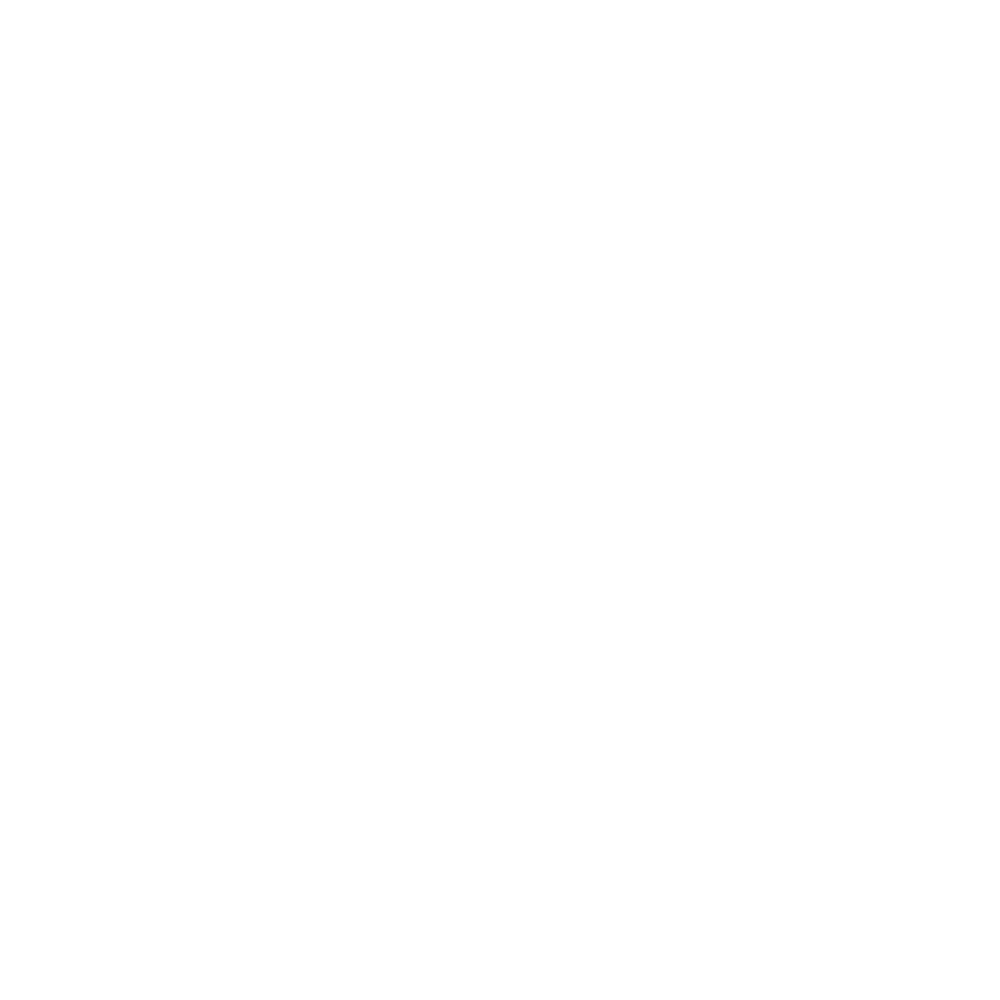
1. insert\_node(Node\* cur, Node\* insert). Функция вставляет переданный узел после определённого элемента списка, на который получает ссылку. Устанавливает предшествующим переданному элементу значение элемента, ссылка на который была передана, а следующим – значение элемента списка, который был следующим для элемента с переданной ссылкой. Ссылки элементов вокруг вставленного также изменяются.
2. find\_value(Node\* head, string value). Функция получает на вход первый узел списка и искомое значение. В цикле перебираюся все элементы списка Если поле элемента равно искомому значению возвращается ссылка на найденный узел. Если узел с искомым значением не найден возвращается nullptr.
3. delete\_node(Node\* cur). Функция получает на вход ссылку на узел который необходимо удалить. Настраиваем связи, как бы пропуская элемент, который нужно удалить и все его поля. В итоге ничего не ссылается на поля удаляемого элемента.
4. print\_node(Node\* node). Выводим данные узла и перенос на следующую строку.
5. print\_list\_back(Node\* head). Начиная с последнего узла, выводим данные из него при помощи print\_node(Node\* node) и переходим к предыдущему узлу, пока не встретим этот узел снова. После цикла выводим перенос на следующую строку.
6. print\_list(Node\* head). Начиная с первого узла, выводим данные из него при помощи print\_node(Node\* node) и переходим к следующему узлу, пока не встретим этот узел снова. После цикла выводим перенос на следующую строку.
7. insert\_task(Node\* head). На вход подаётся первый узел списка. Необходимо ввести данные для нового узла, который будет вставлен в список. Происходит поиск узла в списке с таким же ключевым значением как у нового с помощью функции find\_value(Node\* head, string value). Если такой элемент найден, то новый узел вставляется перед ним. Если же узел не найден, в цикле происходит поиск первого узла с большим значением ключа и новый узел вставляется перед ним.
8. remove\_nums(Node\* head, int remove). На вход подаётся первый узел списка и значение поля по которому будет проходить удаление узлов. В цикле происходит поиск узлов с заданным значениям информационного поля, каждый найденный узел удаляется.
9. make\_new\_list(Node\* head1, Node\* head2) Функция получает на вход два основания разных списков. В 1 списке ищутся элементы со значением информационного поля mark 2, они добавляются во 2 список и удаляются из первого.
   1. Таблица тестов для каждой операции.

Вставка узла в список перед первым узлом с таким же ключом(если такого узла нет, то вставить перед первым узлом, у которого ключ больше).

| Входные данные | Выходные данные |
| --- | --- |
| Номер кассы: 1  Операции:  Код товара: A001  Количество: 5  Единица измерения: шт  Цена за единицу: 10  Сумма к оплате: 50  Дата операции: 2024-04-20  Код товара: B002  Количество: 3  Единица измерения: уп  Цена за единицу: 20  Сумма к оплате: 60  Дата операции: 2024-04-21  Номер кассы: 2  Операции:  Код товара: C003  Количество: 2  Единица измерения: шт  Цена за единицу: 15  Сумма к оплате: 30  Дата операции: 2024-04-22  Номер кассы: 3  Операции:  Код товара: A001  Количество: 10  Единица измерения: шт  Цена за единицу: 10  Сумма к оплате: 100  Дата операции: 2024-04-23  Код товара: B002  Количество: 2  Единица измерения: уп  Цена за единицу: 20  Сумма к оплате: 40  Дата операции: 2024-04-24 | Номер кассы: 1  Операции:  Код товара: A001  Количество: 5  Единица измерения: шт  Цена за единицу: 10  Сумма к оплате: 50  Дата операции: 2024-04-20  Код товара: B002  Количество: 3  Единица измерения: уп  Цена за единицу: 20  Сумма к оплате: 60  Дата операции: 2024-04-21  Номер кассы: 2  Операции:  Код товара: C003  Количество: 2  Единица измерения: шт  Цена за единицу: 15  Сумма к оплате: 30  Дата операции: 2024-04-22  Номер кассы: 3  Операции:  Код товара: A001  Количество: 10  Единица измерения: шт  Цена за единицу: 10  Сумма к оплате: 100  Дата операции: 2024-04-23  Код товара: B002  Количество: 2  Единица измерения: уп  Цена за единицу: 20  Сумма к оплате: 40  Дата операции: 2024-04-24  Номер кассы: 4  Операции: |

Удаление узлов с указанным номером группы

| Входные данные | Выходные данные |
| --- | --- |
| Номер кассы: 1  Операции:  Код товара: A001  Количество: 5  Единица измерения: шт  Цена за единицу: 10  Сумма к оплате: 50  Дата операции: 2024-04-20  Код товара: B002  Количество: 3  Единица измерения: уп  Цена за единицу: 20  Сумма к оплате: 60  Дата операции: 2024-04-21  Номер кассы: 2  Операции:  Код товара: C003  Количество: 2  Единица измерения: шт  Цена за единицу: 15  Сумма к оплате: 30  Дата операции: 2024-04-22  Номер кассы: 3  Операции:  Код товара: A001  Количество: 10  Единица измерения: шт  Цена за единицу: 10  Сумма к оплате: 100  Дата операции: 2024-04-23  Код товара: B002  Количество: 2  Единица измерения: уп  Цена за единицу: 20  Сумма к оплате: 40  Дата операции: 2024-04-24  Номер кассы: 4  Операции: | Номер кассы: 2  Операции:  Код товара: C003  Количество: 2  Единица измерения: шт  Цена за единицу: 15  Сумма к оплате: 30  Дата операции: 2024-04-22  Номер кассы: 3  Операции:  Код товара: A001  Количество: 10  Единица измерения: шт  Цена за единицу: 10  Сумма к оплате: 100  Дата операции: 2024-04-23  Код товара: B002  Количество: 2  Единица измерения: уп  Цена за единицу: 20  Сумма к оплате: 40  Дата операции: 2024-04-24  Номер кассы: 4  Операции: |



Создание нового списка из узлов с оценкой 2

| Входные данные | Выходные данные |
| --- | --- |
| Номер кассы: 2  Операции:  Код товара: C003  Количество: 2  Единица измерения: шт  Цена за единицу: 15  Сумма к оплате: 30  Дата операции: 2024-04-22  Номер кассы: 3  Операции:  Код товара: A001  Количество: 10  Единица измерения: шт  Цена за единицу: 10  Сумма к оплате: 100  Дата операции: 2024-04-23  Код товара: B002  Количество: 2  Единица измерения: уп  Цена за единицу: 20  Сумма к оплате: 40  Дата операции: 2024-04-24  Номер кассы: 4  Операции: | Узел найден. |

1. **Код программы на языке С++**
2. **Linkedlist.h**

#ifndef LINKEDLIST\_H

#define LINKEDLIST\_H

#include <iostream>

#include <string>

**using** **namespace** std;

**struct** Operation {

**string** productCode;

**int** quantity;

**string** unit;

**double** pricePerUnit;

**double** totalAmount;

**string** date;

Operation\* nextOperation; // Указатель на следующую операцию

};

**struct** Node {

**int** cashRegisterNumber;

Operation\* operations; // Указатель на первую операцию

Node\* prev;

Node\* next;

};

**class** LinkedList {

**public**:

LinkedList(); // конструктор

~LinkedList(); // деструктор

**void** createList(); // создание списка

**void** insertNode(**int** cashRegisterNumber); // вставка узла

**void** deleteNode(**int** cashRegisterNumber); // удаление узла

**void** printForward() **const**; // вывод списка слева направо

**void** printBackward() **const**; // вывод списка справа налево

Node\* searchNode(**int** cashRegisterNumber) **const**; // поиск узла по номеру кассы

// Дополнительные операции

**void** addOperation(**int** cashRegisterNumber, **const** Operation& operation); // добавить операцию

**double** calculateIncome(**const** **string**& date) **const**; // определить приход по кассе в указанную дату

**void** cancelOperation(**int** cashRegisterNumber, **const** **string**& productCode); // отменить операцию

**private**:

Node\* head;

Node\* tail;

Node\* createNode(**int** cashRegisterNumber); // создание узла

Operation\* createOperation(**const** Operation& operation); // создание операции

};

#endif

1. **Linkedlist.cpp**

#include "LinkedList.h"

#include <iostream>

LinkedList::LinkedList() {

head = **nullptr**; // Инициализация указателя на голову списка как nullptr

tail = **nullptr**; // Инициализация указателя на хвост списка как nullptr

}

LinkedList::~LinkedList() {

Node\* current = head; // Инициализация указателя current на голову списка

**while** (current != **nullptr**) {

Node\* next = current->next; // Сохранение указателя на следующий узел

**delete** current; // Удаление текущего узла

current = next; // Переход к следующему узлу

}

}

//Создание нового узла

Node\* LinkedList::createNode(**int** cashRegisterNumber) {

Node\* newNode = **new** Node;

newNode->cashRegisterNumber = cashRegisterNumber;

newNode->operations = **nullptr**;

newNode->prev = **nullptr**;

newNode->next = **nullptr**;

**return** newNode;

}

Operation\* LinkedList::createOperation(**const** Operation& operation) {

Operation\* newOperation = **new** Operation; // Выделение памяти под новую операцию

// Инициализация полей новой операции значениями из переданной операции

newOperation->productCode = operation.productCode;

newOperation->quantity = operation.quantity;

newOperation->unit = operation.unit;

newOperation->pricePerUnit = operation.pricePerUnit;

newOperation->totalAmount = operation.totalAmount;

newOperation->date = operation.date;

newOperation->nextOperation = **nullptr**;

**return** newOperation;

}

**void** LinkedList::createList() {

head = **nullptr**;

tail = **nullptr**;

}

**void** LinkedList::insertNode(**int** cashRegisterNumber) {

Node\* newNode = createNode(cashRegisterNumber); // Создание нового узла

**if** (head == **nullptr**) {

head = newNode;

tail = newNode;

}

**else** {

tail->next = newNode;

newNode->prev = tail; // Устанавливаем предыдущим узлом для нового узла текущий хвост

tail = newNode; // Устанавливаем новый узел как хвост списка

}

}

//Удаление узла

**void** LinkedList::deleteNode(**int** cashRegisterNumber) {

Node\* current = head; // Указатель на текущий узел, начиная с головы списка

**while** (current != **nullptr**) {

**if** (current->cashRegisterNumber == cashRegisterNumber) {

**if** (current == head && current == tail) {

head = **nullptr**;

tail = **nullptr**;

}

**else** **if** (current == head) {

head = head->next;

head->prev = **nullptr**;

}

**else** **if** (current == tail) {

tail = tail->prev;

tail->next = **nullptr**;

}

**else** {

current->prev->next = current->next;

current->next->prev = current->prev;

}

**delete** current;

**return**;

}

current = current->next; // Переходим к следующему узлу

}

**cout** << "Касса с номером " << cashRegisterNumber << " не найдена." << **endl**;

}

//Вывод вперед

**void** LinkedList::printForward() **const** {

Node\* current = head; // Указатель на текущий узел, начиная с головы списка

**cout** << "Список слева направо:" << **endl**;

**while** (current != **nullptr**) {

**cout** << "Номер кассы: " << current->cashRegisterNumber << **endl**;

**cout** << "Операции:" << **endl**;

Operation\* operation = current->operations; // Указатель на текущую операцию

**while** (operation != **nullptr**) {

**cout** << "Код товара: " << operation->productCode << **endl**;

**cout** << "Количество: " << operation->quantity << **endl**;

**cout** << "Единица измерения: " << operation->unit << **endl**;

**cout** << "Цена за единицу: " << operation->pricePerUnit << **endl**;

**cout** << "Сумма к оплате: " << operation->totalAmount << **endl**;

**cout** << "Дата операции: " << operation->date << **endl**;

operation = operation->nextOperation; // Переходим к следующей операции

}

current = current->next; // Переходим к следующему узлу

}

}

//Вывод назад

**void** LinkedList::printBackward() **const** {

Node\* current = tail; // Указатель на текущий узел, начиная с хвоста списка

**cout** << "Список справа налево:" << **endl**;

**while** (current != **nullptr**) {

**cout** << "Номер кассы: " << current->cashRegisterNumber << **endl**;

**cout** << "Операции:" << **endl**;

Operation\* operation = current->operations; // Указатель на текущую операцию

**while** (operation != **nullptr**) {

**cout** << "Код товара: " << operation->productCode << **endl**;

**cout** << "Количество: " << operation->quantity << **endl**;

**cout** << "Единица измерения: " << operation->unit << **endl**;

**cout** << "Цена за единицу: " << operation->pricePerUnit << **endl**;

**cout** << "Сумма к оплате: " << operation->totalAmount << **endl**;

**cout** << "Дата операции: " << operation->date << **endl**;

operation = operation->nextOperation; // Переходим к следующей операции

}

current = current->prev; // Переходим к предыдущему узлу

}

}

Node\* LinkedList::searchNode(**int** cashRegisterNumber) **const** {

Node\* current = head; // Указатель на текущий узел, начиная с головы списка

**while** (current != **nullptr**) {

**if** (current->cashRegisterNumber == cashRegisterNumber) {

**return** current;

}

current = current->next;

}

**return** **nullptr**; // Если узел с заданным номером кассы не найден, возвращаем nullptr

}

**void** LinkedList::addOperation(**int** cashRegisterNumber, **const** Operation& operation) {

Node\* node = searchNode(cashRegisterNumber); // Поиск узла с заданным номером кассы

**if** (node != **nullptr**) {

Operation\* newOperation = createOperation(operation);

**if** (node->operations == **nullptr**) {

node->operations = newOperation; // Устанавливаем новую операцию как первую операцию узла

}

**else** { // Если у узла уже есть операции

Operation\* current = node->operations;

**while** (current->nextOperation != **nullptr**) {

current = current->nextOperation;

}

current->nextOperation = newOperation; // Добавляем новую операцию в конец списка операций

}

}

**else** {

**cout** << "Касса с номером " << cashRegisterNumber << " не найдена." << **endl**;

}

}

**double** LinkedList::calculateIncome(**const** **string**& date) **const** {

**double** totalIncome = 0.0; // Переменная для хранения общего прихода

Node\* current = head;

**while** (current != **nullptr**) {

Operation\* operation = current->operations;

**while** (operation != **nullptr**) {

**if** (operation->date == date) {

totalIncome += operation->totalAmount;

}

operation = operation->nextOperation;

}

current = current->next;

}

**return** totalIncome;

}

**void** LinkedList::cancelOperation(**int** cashRegisterNumber, **const** **string**& productCode) {

Node\* node = searchNode(cashRegisterNumber); // Поиск узла с заданным номером кассы

**if** (node != **nullptr**) {

Operation\* current = node->operations;

Operation\* prev = **nullptr**;

**while** (current != **nullptr**) {

**if** (current->productCode == productCode) {

**if** (prev == **nullptr**) {

node->operations = current->nextOperation;

}

**else** {

prev->nextOperation = current->nextOperation;

}

**delete** current;

**cout** << "Операция с кодом товара " << productCode << " отменена для кассы " << cashRegisterNumber << **endl**;

**return**;

}

prev = current;

current = current->nextOperation;

}

**cout** << "Операция с кодом товара " << productCode << " не найдена для кассы " << cashRegisterNumber << **endl**;

}

**else** { // Если узел с заданным номером кассы не найден

**cout** << "Касса с номером " << cashRegisterNumber << " не найдена." << **endl**;

}

}

1. **main.cpp**

#include <iostream>

#include "LinkedList.h"

**using** **namespace** std;

**void** printMenu() {

**cout** << "Меню:" << **endl**;

**cout** << "1. Вывод списка вперед на экран" << **endl**;

**cout** << "2. Вывод списка назад на экран" << **endl**;

**cout** << "3. Вставка нового узла в конец подсписка" << **endl**;

**cout** << "4. Удаление узла" << **endl**;

**cout** << "5. Поиск узла по номеру кассы" << **endl**;

**cout** << "6. Добавить операцию к указанной кассе" << **endl**;

**cout** << "7. Определить приход по кассе в указанную дату" << **endl**;

**cout** << "8. Отменить операцию по указанной кассе и коду товара" << **endl**;

**cout** << "9. Выход из программы" << **endl**;

**cout** << "Введите пункт от 1 до 9: ";

}

**int** **main**() {

**setlocale**(LC\_ALL, "Rus");

LinkedList **list**;

// Создание списка

**list**.createList();

// Добавление касс с операциями

**list**.insertNode(1);

Operation operation1\_1 = { "A001", 5, "шт", 10, 50, "2024-04-20" };

Operation operation1\_2 = { "B002", 3, "уп", 20, 60, "2024-04-21" };

**list**.addOperation(1, operation1\_1);

**list**.addOperation(1, operation1\_2);

**list**.insertNode(2);

Operation operation2\_1 = { "C003", 2, "шт", 15, 30, "2024-04-22" };

**list**.addOperation(2, operation2\_1);

**list**.insertNode(3);

Operation operation3\_1 = { "A001", 10, "шт", 10, 100, "2024-04-23" };

Operation operation3\_2 = { "B002", 2, "уп", 20, 40, "2024-04-24" };

**list**.addOperation(3, operation3\_1);

**list**.addOperation(3, operation3\_2);

**int** choice;

Operation operation;

Node\* foundNode;

**string** dateToCalculateIncome;

**int** cashRegisterNumberToCancelOperation;

**string** productCodeToCancelOperation;

**do** {

printMenu();

**cin** >> choice;

**switch** (choice) {

**case** 1:

**cout** << "Вывод списка вперед:" << **endl**;

**list**.printForward();

**cout** << **endl**;

**break**;

**case** 2:

**cout** << "Вывод списка назад:" << **endl**;

**list**.printBackward();

**cout** << **endl**;

**break**;

**case** 3:

**int** newCashRegisterNumber;

**cout** << "Введите номер кассы для вставки: ";

**cin** >> newCashRegisterNumber;

**list**.insertNode(newCashRegisterNumber);

**break**;

**case** 4:

**int** cashRegisterNumberToDelete;

**cout** << "Введите номер кассы для удаления: ";

**cin** >> cashRegisterNumberToDelete;

**list**.deleteNode(cashRegisterNumberToDelete);

**break**;

**case** 5:

**int** cashRegisterNumberToFind;

**cout** << "Введите номер кассы для поиска: ";

**cin** >> cashRegisterNumberToFind;

foundNode = **list**.searchNode(cashRegisterNumberToFind);

**if** (foundNode != **nullptr**) {

**cout** << "Узел найден." << **endl**;

}

**else** {

**cout** << "Узел не найден." << **endl**;

}

**break**;

**case** 6:

**int** cashRegisterNumberToAddOperation;

**cout** << "Введите номер кассы для добавления операции: ";

**cin** >> cashRegisterNumberToAddOperation;

**cout** << "Введите данные операции:" << **endl**;

**cout** << "Код товара: ";

**cin** >> operation.productCode;

**cout** << "Количество: ";

**cin** >> operation.quantity;

**cout** << "Единица измерения: ";

**cin** >> operation.unit;

**cout** << "Цена за единицу: ";

**cin** >> operation.pricePerUnit;

**cout** << "Сумма к оплате: ";

**cin** >> operation.totalAmount;

**cout** << "Дата операции: ";

**cin** >> operation.date;

**list**.addOperation(cashRegisterNumberToAddOperation, operation);

**break**;

**case** 7:

**cout** << "Введите дату для определения прихода: ";

**cin** >> dateToCalculateIncome;

**cout** << "Приход по кассе на дату " << dateToCalculateIncome << " составляет " << **list**.calculateIncome(dateToCalculateIncome) << **endl**;

**break**;

**case** 8:

**cout** << "Введите номер кассы для отмены операции: ";

**cin** >> cashRegisterNumberToCancelOperation;

**cout** << "Введите код товара для отмены операции: ";

**cin** >> productCodeToCancelOperation;

**list**.cancelOperation(cashRegisterNumberToCancelOperation, productCodeToCancelOperation);

**break**;

**case** 9:

**cout** << "Выход из программы." << **endl**;

**break**;

**default**:

**cout** << "Неверный ввод. Попробуйте еще раз." << **endl**;

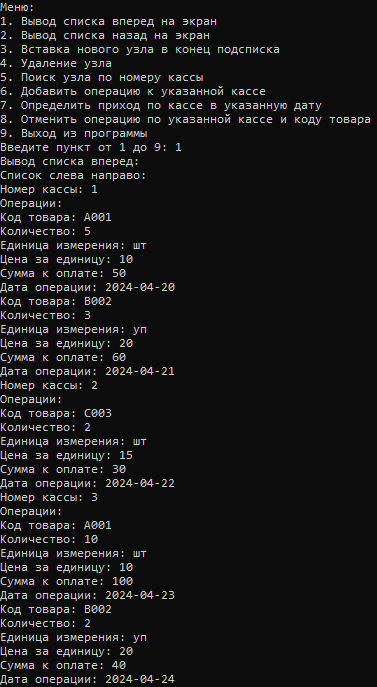
}

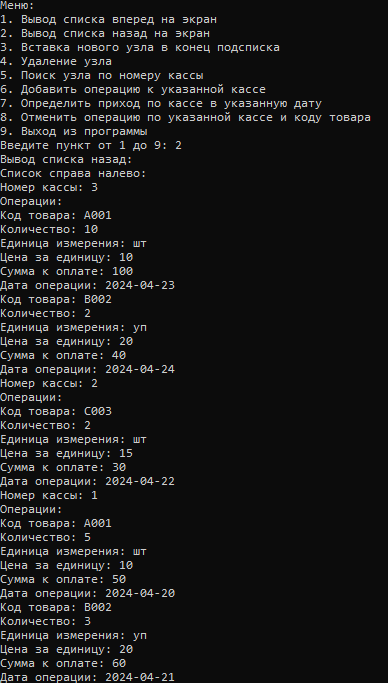
} **while** (choice != 9);

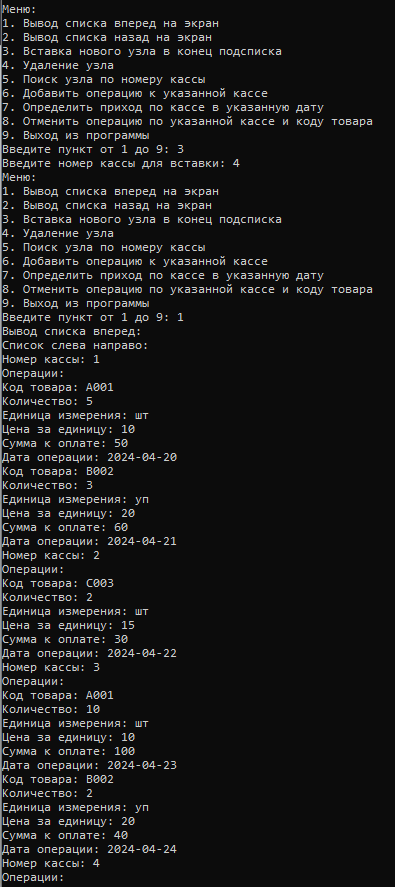
**return** 0;

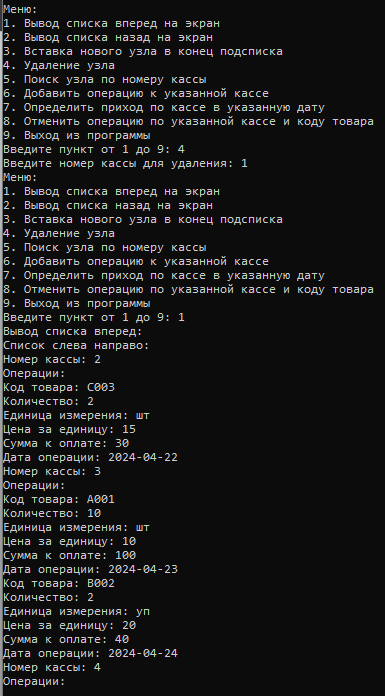
}

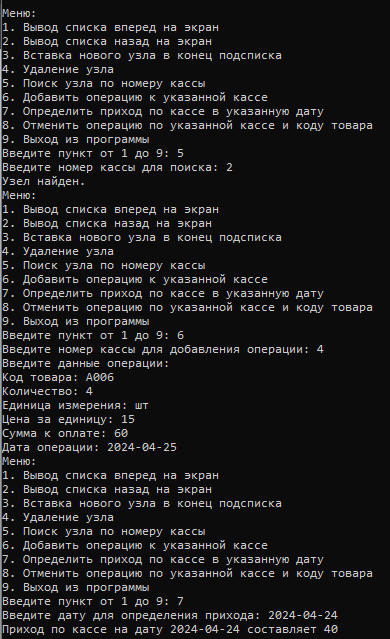
1. **Результаты тестирования**

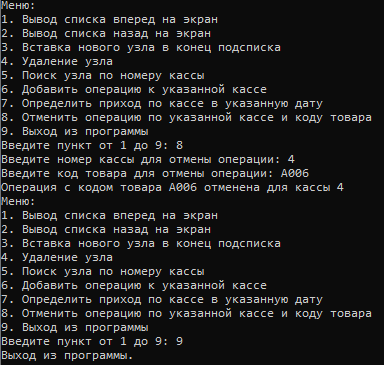












1. **Вывод**

В процессе выполнения данной практической работы была разработана многомодульная программа на языке C++, демонстрирующая работу с линейным двунаправленным динамическим списком. Программа включает в себя реализацию основных операций над списком, таких как создание, вставка и удаление узлов, вывод списка в обоих направлениях, а также поиск узла по заданному значению.

Эта практическая работа позволила ознакомиться с основами управления двунаправленным динамическим списком в языке программирования C++ и приобрести практические навыки работы с указателями, структурами и операциями над ними. Полученный опыт будет полезен при решении задач, связанных с обработкой и управлением структурированными данными в программах на C++.

1. **Литература**

1. Бхаргава А. Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. – СПб: Питер, 2017. – 288 с.

2. Вирт Н. Алгоритмы + структуры данных = программы. – М.: Мир, 1985. – 406 с.

3. Кнут Д.Э. Искусство программирования, том 3. Сортировка и поиск, 2-е изд. – М.: ООО «И.Д. Вильямс», 2018. – 832 с.

4. Кораблин Ю.П. Структуры и алгоритмы обработки данных: учебно-методическое пособие / Ю.П. Кораблин, В.П. Сыромятников, Л.А. Скворцова. – М.: РТУ МИРЭА, 2020. — 219 с.

5. Кормен Т.Х. и др. Алгоритмы: построение и анализ, 3-е изд. – М.: ООО «И.Д.Вильямс», 2013. – 1328 с.

6. Макконнелл Дж. Основы современных алгоритмов. Активный обучающий метод. 3-е доп. изд., - М.: Техносфера, 2018. – 416 с.

7. Седжвик Р. Фундаментальные алгоритмы на C++. Анализ/Структуры данных/Сортировка/Поиск. – К.: Издательство «Диасофт», 2001. – 688 с.

8. Скиена С. Алгоритмы. Руководство по разработке, - 2-е изд. – СПб: БХВ-Петербург, 2011. – 720 с.

9. Хайнеман Д. и др. Алгоритмы. Справочник с примерами на C, C++, Java и Python, 2-е изд. – СПб: ООО «Альфа-книга», 2017. – 432 с.

10. AlgoList – алгоритмы, методы, исходники [Электронный ресурс]. URL: http://algolist.manual.ru/ (дата обращения 15.03.2022).

11. Алгоритмы – всё об алгоритмах / Хабр [Электронный ресурс]. URL: https://habr.com/ru/hub/algorithms/ (дата обращения 15.03.2022).

12. НОУ ИНТУИТ | Технопарк Mail.ru Group: Алгоритмы и структуры данных [Электронный ресурс]. URL: https://intuit.ru/studies/courses/3496/738/info (дата обращения 15.03.2022).