МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение

высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания №1.6

Тема:

Двунаправленные динамические списки

Дисциплина: «Структуры и алгоритмы обработки данных»

Выполнил студент: Враженко Д.О.

Группа: ИКБО-10-23

Вариант: 7

Москва – 2024

ЦЕЛЬ РАБОТЫ

Получение знаний и практических навыков управления двунаправленным списком в программах на языке C++.

ХОД РАБОТЫ

1. Постановка задачи

Разработать многомодульную программу, которая демонстрирует выполнение всех операций, определенных вариантом, 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. Разработать 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).

Индивидуальный вариант 7:

Тип информационной части узла списка: Код товара (буквенно-цифровой), дата продажи, цена, отметка о возврате.

Дополнительные операции:

• отсортировать список, располагая элементы в хронологическом порядке.

• удалить все узлы по заданному товару, проданному в указанную дату.

• сформировать новый список, из узлов исходного, по тем товарам, по которым имеется возврат.

2. Список операций над списком

• вставка узла в список;

• удаление узла из списка;

• вывод списка слева направо;

• вывод списка справа налево;

• поиск узла в списке;

• сортировка списка по дате продажи;

• удаление всех узлов с заданными товаром и датой продажи;

• формирование нового списка по товарам с возвратом.

2.1. Структура узла двунаправленного списка

Узел содержит 6 полей: productCode, saleDate, price, returnMark, prev, next.

Поля строкового типа:

• productCode – код товара (буквенно-цифровой).

• saleDate – дата продажи.

Поле вещественного типа:

• price – цена товара.

Поле логического типа:

• returnMark – отметка о возврате.

Поля типа указателя на узел:

• prev – указатель на предыдущий узел в списке.

• next – указатель на следующий узел в списке.

2.2. Рисунки базовых операций полученного списка

На рис. 1 представлен рисунок базовой операции вставки узла в список.

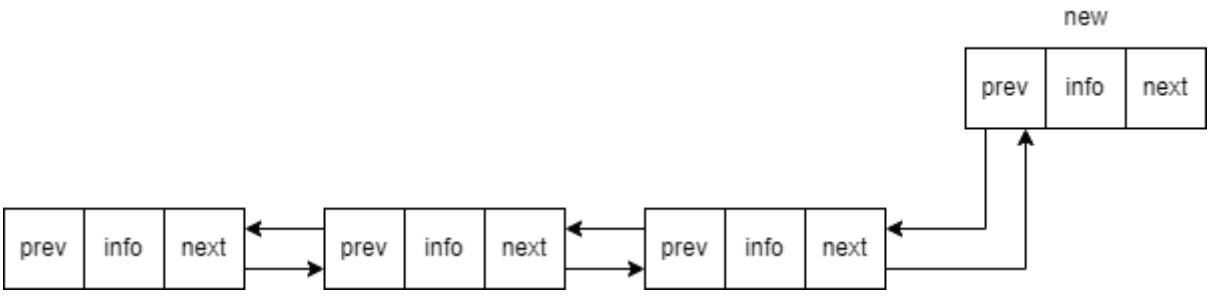

Рисунок 1 - Базовая операция вставки узла в список

На рис. 2 представлен рисунок базовой операции удаления узла из списка.

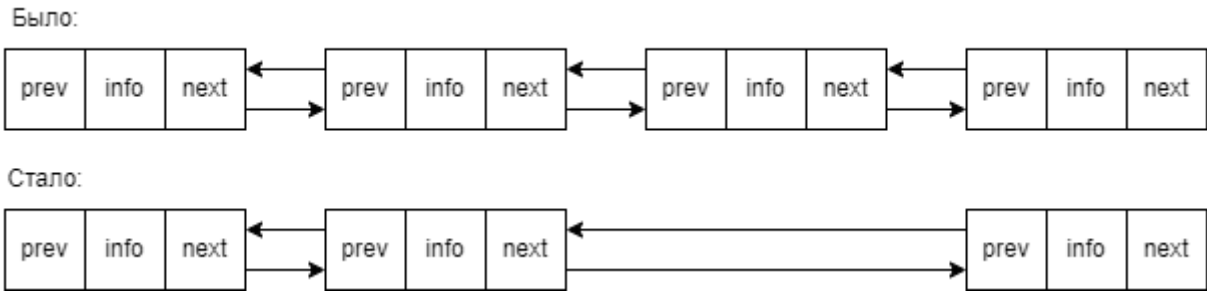

Рисунок 2 - Базовая операция удаления узла из списка

На рис. 3 представлен рисунок базовой операции вывода списка слева направо.

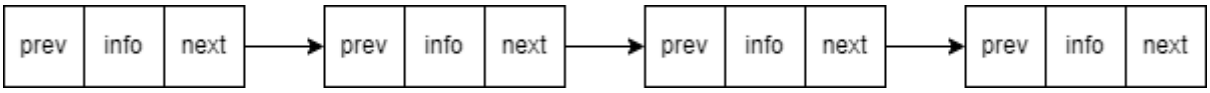

Рисунок 3 - Базовая операция вывода списка слева направо

На рис. 4 представлен рисунок базовой операции вывода списка справа налево.

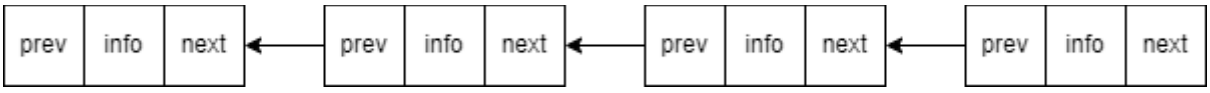

Рисунок 4 - Базовая операция вывода списка справа налево

На рис. 5 представлен рисунок базовой операции поиска узла в списке.

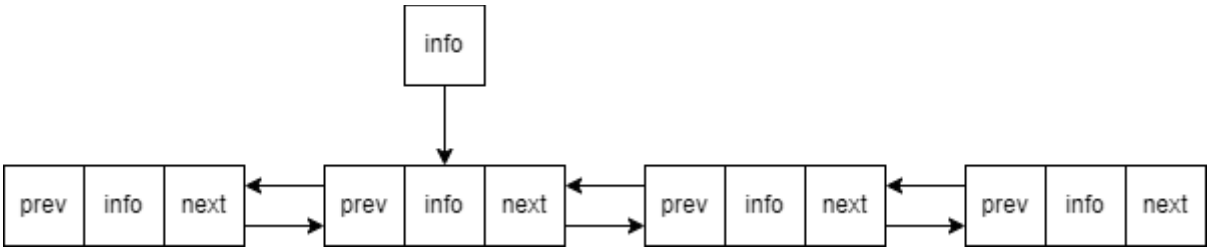

Рисунок 5 - Базовая операция поиска узла в списке

2.3. Изображение структуры данных, используемой в операциях

На рис. 6 представлен рисунок структуры данных, используемой в операциях (базовых и дополнительных).

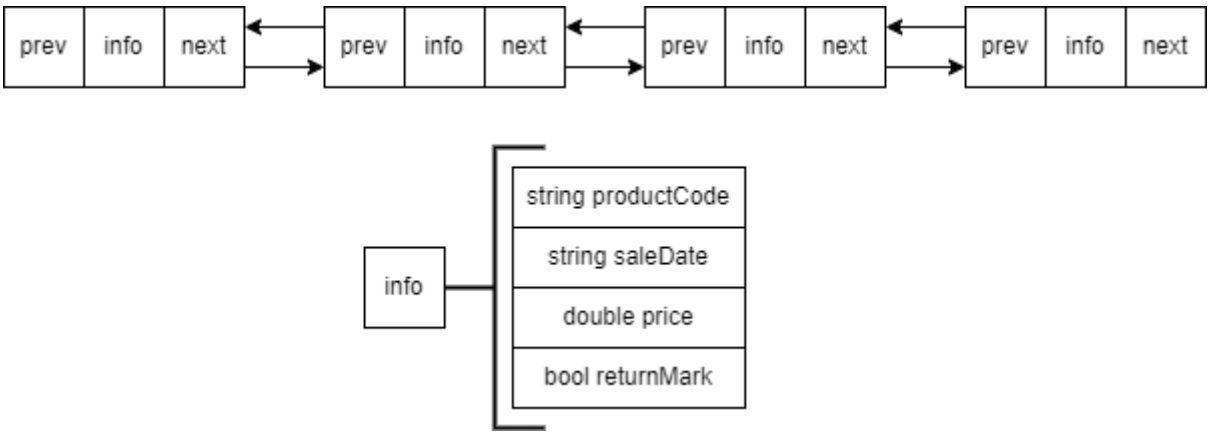

Рисунок 6 - Структура данных, используемая в операциях

2.4. Словесное описание алгоритма

createList: Возврат значения nullptr.

insertNode: Сначала создается новый узел и заполняются его поля значениями переданных параметров. Затем проверяется, существует ли уже список (если tail не равен nullptr), и при необходимости устанавливается ссылка на новый узел. Далее устанавливаются ссылки на предыдущий и следующий узлы в соответствии с порядком в списке. Если список пустой (head равен nullptr), то устанавливается новый узел как начальный узел списка.

deleteNode: В начале метода создается указатель current, который указывает на начало списка. Далее происходит цикл, в котором проверяется каждый узел списка. Если код товара текущего узла совпадает с заданным кодом, то выполняется удаление узла:

1. Если у текущего узла есть предыдущий узел, то перенаправляется ссылка с предыдущего узла на следующий узел текущего узла.

2. Если у текущего узла есть следующий узел, то перенаправляется ссылка с следующего узла на предыдущий узел текущего узла.

3. Если текущий узел был начальным узлом списка, то обновляется указатель на начало списка, указывая на следующий узел текущего узла.

4. Удаляется текущий узел.

5. Возвращается из метода.

printListForward: В начале метода создается указатель current, который указывает на начало списка. Далее запускается цикл, в котором происходит вывод информации о каждом узле списка:

1. Выводится информация о коде товара, дате продажи, цене и отметке о возврате текущего узла с помощью оператора вывода cout.

2. При выводе отметки о возврате используется тернарный оператор для вывода "Да", если отметка равна true, и "Нет", если отметка равна false.

3. Переход к следующему узлу списка путем обновления указателя current на следующий узел.

printListBackward: В начале метода создается указатель current, который указывает на конец списка. Далее запускается цикл, в котором происходит вывод информации о каждом узле списка:

1. Выводится информация о коде товара, дате продажи, цене и отметке о возврате текущего узла с помощью оператора вывода cout.

2. При выводе отметки о возврате используется тернарный оператор для вывода "Да", если отметка равна true, и "Нет", если отметка равна false.

3. Переход к предыдущему узлу списка путем обновления указателя current на предыдущий узел.

searchNode: В начале метода создается указатель current, который указывает на начало списка. Далее запускается цикл, в котором происходит проверка каждого узла списка:

1. Если код товара текущего узла совпадает с заданным кодом, то метод возвращает указатель на текущий узел.

2. Иначе указатель current переходит к следующему узлу списка.

Если весь список пройден и узел с заданным кодом не найден, метод возвращает nullptr.

compareBySaleDate: В начале метода создаются переменные time1 и time2 типа tm для хранения информации о времени (дате продажи) из узлов node1 и node2 соответственно. Далее используется объект stringstream для парсинга

строки даты продажи из узлов и заполнения структуры tm с помощью функции get_time.

Затем происходит сравнение дат продажи узлов:

1. Сначала сравниваются года (time1.tm_year) узлов. Если год у первого узла меньше, чем у второго, то возвращается true, что означает, что первый узел должен стоять перед вторым в списке по времени продажи.

2. Если года одинаковы, то сравниваются месяцы (time1.tm_mon). Если месяц у первого узла меньше, чем у второго, то возвращается true.

3. В случае, если месяцы также одинаковы, сравниваются дни (time1.tm_mday). Возвращается true, если день у первого узла меньше, чем у второго.

В конечном итоге, метод возвращает результат сравнения дат продажи узлов, опираясь на день, месяц и год.

sortListBySaleDate: В начале метода выполняется проверка наличия элементов в списке: если список пустой или содержит только один элемент, сортировка не требуется, поэтому метод завершает свою работу.

Затем создается вектор nodeVector, в котором будут храниться указатели на все узлы списка. Затем указатель current устанавливается на начало списка, и происходит перебор всех элементов списка с добавлением указателей на узлы в вектор nodeVector.

Далее вектор nodeVector сортируется с использованием функции std::sort и функции сравнения compareBySaleDate, которая была представлена ранее. После сортировки указатель head переустанавливается на первый элемент отсортированного вектора, и устанавливается связь между отсортированными узлами.

В цикле происходит установка связей между соседними узлами, что позволяет установить правильный порядок элементов в списке. Наконец, последний элемент списка соединяется с nullptr, чтобы завершить цепочку узлов.

deleteNodesByProductAndDate: В начале метода указатель current устанавливается на начало списка. Затем выполняется цикл, в котором происходит про-

верка каждого узла списка. Для каждого узла проверяется соответствие productCode и saleDate заданным значениям.

Если текущий узел соответствует заданным критериям, то:

1. Устанавливаются связи между соседними узлами таким образом, чтобы убрать текущий узел из списка. Если текущий узел имеет предыдущий элемент (current->prev != nullptr), то предыдущий узел указывает на следующий узел удаляемого узла (current->next). Если текущий узел имеет следующий элемент (current->next != nullptr), то следующий узел указывает на предыдущий узел удаляемого узла (current->prev).

2. Если текущий узел является началом списка (current == head), то указатель head переустанавливается на следующий узел после удаляемого узла.

3. Удаляется текущий узел.

После завершения цикла, все узлы, которые соответствуют заданным критериям, будут удалены из списка, а связи между оставшимися узлами будут обновлены с учетом удаленных узлов.

createReturnList: В начале метода инициализируются указатели returnHead и returnTail как nullptr, которые будут указывать на начало и конец нового списка соответственно. Также инициализируется указатель current на начало исходного списка.

Затем выполняется цикл, в котором каждый узел текущего списка проверяется на наличие пометки возврата. Если у текущего узла есть пометка возврата, то происходит создание нового узла (newNode), копирование информации из текущего узла в новый узел и установка связей между узлами нового списка.

1. Создается новый узел (newNode) и копируются данные о продукт коде (productCode), дате продажи (saleDate), цене (price) и пометке возврата (returnMark) из текущего узла.

2. Устанавливаются связи между узлами нового списка: предыдущий узел (prev) нового узла указывает на текущий хвост (returnTail), а следующий узел (next) нового узла устанавливается в nullptr.

3. Если returnHead равен nullptr, то новый узел становится началом нового списка, иначе предыдущий хвост (returnTail) устанавливает следующий узел на новый узел.

4. Хвост нового списка (returnTail) обновляется на новый узел.

По завершении цикла метод возвращает указатель на начало нового списка, который содержит все узлы с пометкой возврата из исходного списка.

2.5. Таблица тестов

Операция	"Входные" данные	"Выходные" данные
Вставить узел	123ABC 20.11.2015 19.99 0	123ABC 20.11.2015 19.99 0
Вставить узел	2CDFGH 17.06.2010 10.99 1	123ABC 20.11.2015 19.99 0 2CDFGH 17.06.2010 10.99 1
Вставить узел	34OP54 05.12.2018 2.99 1	123ABC 20.11.2015 19.99 0 2CDFGH 17.06.2010 10.99 1 34OP54 05.12.2018 2.99 1
Удалить узел	2CDFGH	123ABC 20.11.2015 19.99 0 34OP54 05.12.2018 2.99 1
Вставить узел	8G7TN4 11.05.2015 4.49 1	123ABC 20.11.2015 19.99 0 34OP54 05.12.2018 2.99 1 8G7TN4 11.05.2015 4.49 1
Вставить узел	4AA400 08.11.2022 17.49 0	123ABC 20.11.2015 19.99 0 34OP54 05.12.2018 2.99 1 8G7TN4 11.05.2015 4.49 1 4AA400 08.11.2022 17.49 0
Вывести список слева направо		123ABC 20.11.2015 19.99 0 34OP54 05.12.2018 2.99 1 8G7TN4 11.05.2015 4.49 1 4AA400 08.11.2022 17.49 0
Вывести список		4AA400 08.11.2022 17.49 0

справа налево		8G7TN4 11.05.2015 4.49 1 34OP54 05.12.2018 2.99 1 123ABC 20.11.2015 19.99 0
Поиск узла по коду товара	34OP54	34OP54 05.12.2018 2.99 1
Сортировать список по дате продажи		
Вывести список слева направо		8G7TN4 11.05.2015 4.49 1 123ABC 20.11.2015 19.99 0 34OP54 05.12.2018 2.99 1 4AA400 08.11.2022 17.49 0
Вставить узел	8G7TN4 11.05.2015 4.49 1	8G7TN4 11.05.2015 4.49 1 123ABC 20.11.2015 19.99 0 34OP54 05.12.2018 2.99 1 4AA400 08.11.2022 17.49 0 8G7TN4 11.05.2015 4.49 1
Удалить все узлы с заданным товаром и датой продажи	8G7TN4 11.05.2015	
Вывести список слева направо		123ABC 20.11.2015 19.99 0 34OP54 05.12.2018 2.99 1 4AA400 08.11.2022 17.49 0
Сформировать новый список по товарам с возвратом		34OP54 05.12.2018 2.99 1

3. Код программы

Файл "main.cpp":

```
#include "struct.h"
#include "main_functions.cpp"
#include "additional_functions.cpp"
using namespace std;

int main()
{
```

```
	setlocale(LC_ALL, "Russian");
	Node* head = nullptr;
	Node* tail = nullptr;
	Node* returnListHead = nullptr;
	int choice;
	string main_productCode, main_saleDate; double main_price; bool main_returnMark;
	do
	{
		cout << "\nОперации\n"
			<< " 1. Вставить узел\n"
			<< " 2. Удалить узел\n"
			<< " 3. Вывести список слева направо\n"
			<< " 4. Вывести список справа налево\n"
			<< " 5. Поиск узла по коду товара\n"
			<< " 6. Сортировать список по дате продажи\n"
			<< " 7. Удалить все узлы с заданным товаром и датой продажи\n"
			<< " 8. Сформировать новый список по товарам с возвратом\n"
			<< " 0. Выйти из программы\n"
			<< "Выберите действие: ";
		cin >> choice;
		switch (choice)
		{
		case 1:
			cout << "Введите данные нового узла:\n";
			cout << "Код товара (буквенно-цифровой): ";
			cin >> main_productCode;
			cout << "Дата продажи: ";
			cin >> main_saleDate;
			cout << "Цена: ";
			cin >> main_price;
			cout << "Отметка о возврате (0 - Нет, 1 - Да): ";
			cin >> main_returnMark;
			insertNode(head, tail, main_productCode, main_saleDate, main_price, main_returnMark);
			break;
		case 2:
			cout << "Введите код товара узла для удаления: ";
			cin >> main_productCode;
			deleteNode(head, main_productCode);
			break;
```

```
                case 3:
                        cout << "Список слева направо:\n";
                        printListForward(head);
                        break;
                case 4:
                        cout << "Список справа налево:\n";
                        printListBackward(tail);
                        break;
                case 5:
                        cout << "Введите код товара для поиска: ";
                        cin >> main_productCode;
                        if (Node* foundNode = searchNode(head, main_productCode))
                        {
                                cout << "Узел найден:\n";
                                cout << foundNode->productCode << ", " << foundNode-
>saleDate << ", " << foundNode->price << ", " << (foundNode->returnMark ? "Да" :
"Нет") << endl;
                        }
                        else cout << "Узел с указанным кодом товара не найден.\n";
                        break;
                case 6:
                        sortListBySaleDate(head);
                        cout << "Список отсортирован по дате продажи.\n";
                        break;
                case 7:
                        cout << "Введите код товара и дату продажи для удаления
соответствующих узлов:\n";
                        cout << "Код товара: ";
                        cin >> main_productCode;
                        cout << "Дата продажи: ";
                        cin >> main_saleDate;
                        deleteNodesByProductAndDate(head,           main_productCode,
main_saleDate);
                        break;
                case 8:
                        returnListHead = createReturnList(head);
                        cout << "Сформирован новый список из узлов с возвратом:\n";
                        printListForward(returnListHead);
                        while (returnListHead != nullptr)
                        {
                                Node* nextNode = returnListHead->next;
                                delete returnListHead;
```

```
                        returnListHead = nextNode;
                    }
                    break;
                case 0:
                    cout << "Выход из программы.\n";
                    break;
                default:
                    cout << "Некорректный ввод. Попробуйте еще раз.\n";
                }
            } while (choice != 0);
            while (head != nullptr)
            {
                Node* nextNode = head->next;
                delete head;
                head = nextNode;
            }
            return (0);
        }
```

Файл "struct.h":

```
        #ifndef _STRUCT_H
        #define _STRUCT_H
        #include <iostream>
        #include <string>
        using namespace std;

        struct Node
        {
            string productCode;
            string saleDate;
            double price;
            bool returnMark;
            Node* prev = nullptr;
            Node* next = nullptr;
        };

        Node* createList();
        void insertNode(Node*& head, const string& productCode, const string& saleDate,
double price, bool returnMark);
        void deleteNode(Node*& head, const string& productCode);
        void printListForward(const Node* head);
        void printListBackward(const Node* tail);
        Node* searchNode(const Node* head, const string& productCode);
```

```
#endif
```

Файл "main_functions.cpp":

```
#include "struct.h"

Node* createList()
{ return nullptr; }

void insertNode(Node*& head, Node*& tail, string& productCode, string& saleDate,
double price, bool returnMark)
{
	Node* newNode = new Node;
	newNode->productCode = productCode;
	newNode->saleDate = saleDate;
	newNode->price = price;
	newNode->returnMark = returnMark;
	newNode->prev = tail;
	newNode->next = nullptr;
	if (tail != nullptr)
		tail->next = newNode;
	tail = newNode;
	if (head == nullptr)
		head = newNode;
}

void deleteNode(Node*& head, string& productCode)
{
	Node* current = head;
	while (current != nullptr)
	{
		if (current->productCode == productCode)
		{
			if (current->prev != nullptr)
				current->prev->next = current->next;
			if (current->next != nullptr)
				current->next->prev = current->prev;
			if (current == head)
				head = current->next;
			delete current;
			return;
		}
		current = current->next;
```

```
        }
    }

    void printListForward(Node* head)
    {
        Node* current = head;
        while (current != nullptr)
        {
            cout << current->productCode << ", " << current->saleDate << ", "
<< current->price << ", " << (current->returnMark ? "Да" : "Нет") << endl;
            current = current->next;
        }
    }

    void printListBackward(Node* tail)
    {
        Node* current = tail;
        while (current != nullptr)
        {
            cout << current->productCode << ", " << current->saleDate << ", "
<< current->price << ", " << (current->returnMark ? "Да" : "Нет") << endl;
            current = current->prev;
        }
    }

    Node* searchNode(Node* head, string& productCode)
    {
        Node* current = head;
        while (current != nullptr)
        {
            if (current->productCode == productCode)
                return const_cast<Node*>(current);
            current = current->next;
        }
        return nullptr;
    }
```

Файл "additional_functions.cpp":

```
    #include "struct.h"
    #include <vector>
    #include <algorithm>
    #include <ctime>
    #include <sstream>
```

```
#include <iomanip>

bool compareBySaleDate(Node* node1, Node* node2)
{
	tm time1 = {};
	stringstream ss1(node1->saleDate);
	ss1 >> get_time(&time1, "%d.%m.%Y");
	tm time2 = {};
	stringstream ss2(node2->saleDate);
	ss2 >> get_time(&time2, "%d.%m.%Y");
	if (time1.tm_year != time2.tm_year)
		return time1.tm_year < time2.tm_year;
	if (time1.tm_mon != time2.tm_mon)
		return time1.tm_mon < time2.tm_mon;
	return time1.tm_mday < time2.tm_mday;
}

void sortListBySaleDate(Node*& head)
{
	if (head == nullptr || head->next == nullptr)
		return;
	vector<Node*> nodeVector;
	Node* current = head;
	while (current != nullptr)
	{
		nodeVector.push_back(current);
		current = current->next;
	}
	sort(nodeVector.begin(), nodeVector.end(), compareBySaleDate);
	head = nodeVector[0];
	head->prev = nullptr;
	for (size_t i = 1; i < nodeVector.size(); ++i)
	{
		nodeVector[i]->prev = nodeVector[i - 1];
		nodeVector[i - 1]->next = nodeVector[i];
	}
	nodeVector.back()->next = nullptr;
}

void deleteNodesByProductAndDate(Node*& head, string& productCode, string&
saleDate)
{
```

```
		Node* current = head;
		while (current != nullptr)
		{
			Node* nextNode = current->next;
			if (current->productCode == productCode && current->saleDate == saleDate)
			{
				if (current->prev != nullptr)
					current->prev->next = current->next;
				if (current->next != nullptr)
					current->next->prev = current->prev;
				if (current == head)
					head = current->next;
				delete current;
			}
			current = nextNode;
		}
	}

	Node* createReturnList(Node* head)
	{
		Node* returnHead = nullptr;
		Node* returnTail = nullptr;
		Node* current = head;
		while (current != nullptr)
		{
			if (current->returnMark)
			{
				Node* newNode = new Node;
				newNode->productCode = current->productCode;
				newNode->saleDate = current->saleDate;
				newNode->price = current->price;
				newNode->returnMark = current->returnMark;
				newNode->prev = returnTail;
				newNode->next = nullptr;
				if (returnHead == nullptr)
					returnHead = newNode;
				else
					returnTail->next = newNode;
				returnTail = newNode;
			}
			current = current->next;
```

```
        }
        return returnHead;
}
```

4. Выводы

Изучен двунаправленный список и реализован на языке C++. Разработаны функции для работы с этой структурой. Протестирована их работоспособность.