|  |
| --- |
| 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: Елисеев И.А. |
|  | Группа: ИКБО-74-23 |

Москва – 2024

СОДЕРЖАНИЕ

[1 ЦЕЛЬ 3](#_gjdgxs)

[2 ЗАДАНИЕ 4](#_30j0zll)

[2.1 Формулировка задачи 4](#_1fob9te)

[2.2. Определение структуры узла двунаправленного списка 5](#_2et92p0)

[2.3 Вставка узла в начале списка 6](#_3dy6vkm)

[2.4 Удаление узла 8](#_1t3h5sf)

[2.5 Вывод элементов списка на экран. 1](#_4d34og8)

[2.6 Поиск узла с заданным значением. 1](#_2s8eyo1)

[2.7 Упорядочить созданный список из n узлов по стране изготовителю. 1](#_17dp8vu)

[2.8 Установить дату продажи автомобиля. 1](#_3rdcrjn)

[2.9 Удалить все узлы по проданным автомобилям. 1](#_26in1rg)

[2.10 Реализация алгоритма на языке C++ и проведение тестирования 1](#_35nkun2)

[2.10.1 Реализация на языке программирования C++ 1](#_1ksv4uv)

[2.10.2 Тестирование 1](#_44sinio)

[3 ВЫВОДЫ 1](#_3j2qqm3)

[4 ЛИТЕРАТУРА 1](#_1y810tw)

# **1 ЦЕЛЬ**

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

# **2 ЗАДАНИЕ**

## 2.1 Формулировка задачи

В списке №10, Вариант 10

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. 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.

6. Оформите отчет по разработке программы в соответствии с требованиями задания по однонаправленному списку.

Примечание: в определении информационной части узла варианта, подчеркнутое поле считать полем ключа.

**Индивидуальный вариант**. Тип информационной части узла: Марка автомобиля, страна изготовитель, год выпуска, цена. Дата продажи (заполняется не сразу).

**Дополнительные операции:**

Упорядочить созданный список из n узлов так, чтобы узлы были упорядочены по стране изготовителю (будут сформированы подсписки по стране).

Вставить новый узел со сведениями об автомобиле какой-то страны в начало своего подсписка.

Установить дату продажи проданному автомобилю.

Удалить все узлы по проданным автомобилям.

## 2.2. Определение структуры узла двунаправленного списка

Определим структуру узла двунаправленного списка согласно варианту.

Структура Node представляет собой элемент двунаправленного списка машин. В данной структуре содержатся следующие поля: переменная brand типа string, которая хранит марку автомобиля, переменная country типа string, которая хранит страну производителя автомобиля, переменная year типа int, которая хранит год выпуска автомобиля, переменная price типа double, которая хранит цену автомобиля, переменная Date типа string, которая хранит дату продажи автомобиля, указатель prev типа Node, который указывает на предыдущий элемент в списке, указатель next типа Node, который указывает на следующий элемент в списке. Конструктор Node принимает значения всех полей структуры и инициализирует соответствующие переменные. Указатели prev и next при создании элемента списка присваиваются значению nullptr.

Данное описание структуры представлено в виде кода на C++ в блоке кода 1, а его изображение на рисунке 1.

| struct Node {  string brand;  string country;  int year;  double price;  string Date;  Node\* prev;  Node\* next; }; |
| --- |

Листинг 1 - Структура

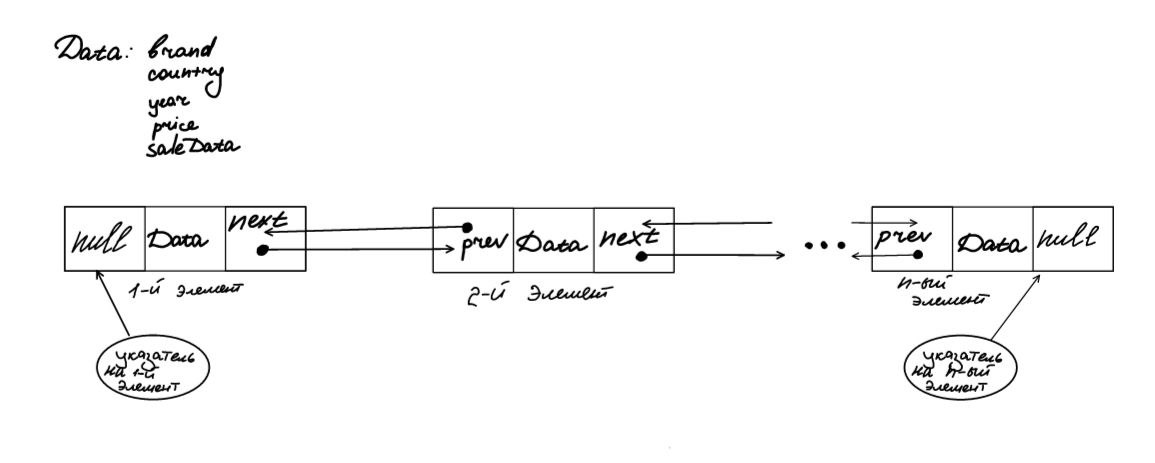


Рисунок 1 - Изображение структуры данных

## 2.3 Вставка узла в начале списка

Отобразим выполнение данной операции на рисунке 2.

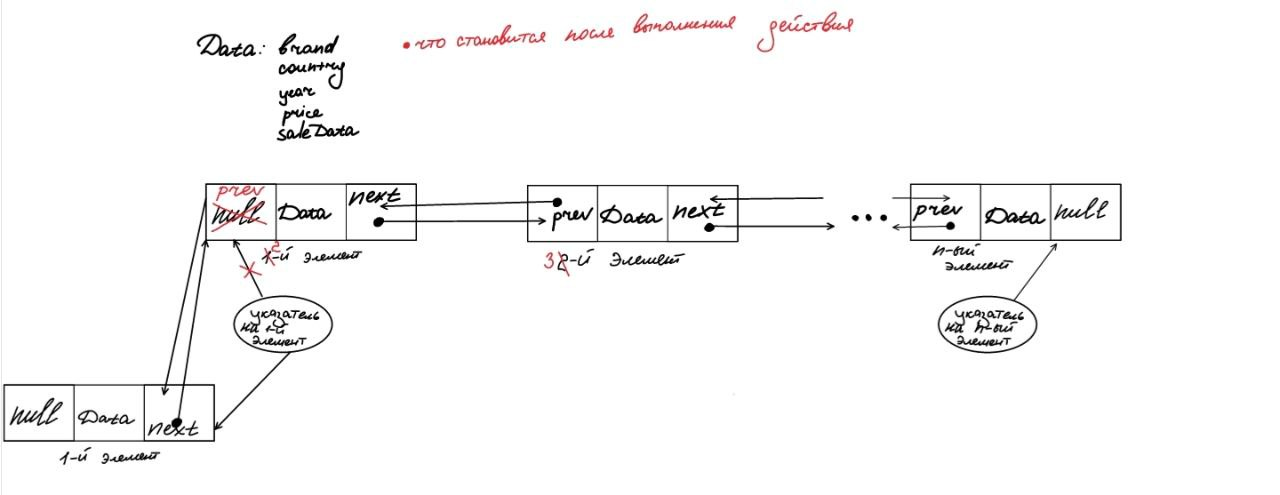


Рисунок 2 - Изображение добавление нового узла

Метод insertAtBeginning предназначен для вставки нового узла в начало подсписка. Он создает новый узел и инициализирует его поля так же, как и в предыдущем методе. Затем метод проверяет, пуст ли список. Если да, новый узел становится как головным, так и хвостовым элементом списка. Если список не пуст, метод ищет первый узел с такой же страной производителем автомобиля и вставляет новый узел перед ним. Если в списке нет узла с такой страной, новый узел присоединяется к концу списка.

Реализация данного алгоритма представлена в блок коде 9. Данные для тестирования будут приведены в таблице 7.

| // Вставить узел в начало подсписка void List::insertAtBeginning(string brand, string country, int year, double price) {  Node\* newNode = new Node;  newNode->brand = brand;  newNode->country = country;  newNode->year = year;  newNode->price = price;  newNode->Date = ""; // Пока не продан   // Если список пуст, вставляем узел как головной  if (head == nullptr) {  newNode->prev = nullptr;  newNode->next = nullptr;  head = newNode;  tail = newNode;  }  else {  // Находим первый узел с той же страной изготовителем и вставляем перед ним  Node\* current = head;  while (current != nullptr && current->country == country)  current = current->next;   if (current == head) {  // Вставляем перед текущим головным узлом  newNode->prev = nullptr;  newNode->next = head;  head->prev = newNode;  head = newNode;  }  else if (current == nullptr) {  // Если такой страны нет в списке, вставляем в конец  newNode->prev = tail;  newNode->next = nullptr;  tail->next = newNode;  tail = newNode;  }  else {  // Вставляем перед узлом с той же страной изготовителем  newNode->prev = current->prev;  newNode->next = current;  current->prev->next = newNode;  current->prev = newNode;  }  } } |
| --- |

Листинг 2 - Добавление нового узла в начале списка

Таблица 1 - Данные для тестирования

| № | Входные данные | Добавить | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Бренд: Ford, Страна: США, Год: 2016, Цена: 18000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Honda, Страна: Япония, Год: 2020, Цена: 22000  Бренд: BMW, Страна: Германия, Год: 2018, Цена: 35000  Бренд: Toyota, Страна: Япония, Год: 2019, Цена: 25000 | Audi, Германия, 2021, 45000.0 | Бренд: Audi, Страна: Германия, Год: 2021, Цена: 45000.0  Бренд: Ford, Страна: США, Год: 2016, Цена: 18000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Honda, Страна: Япония, Год: 2020, Цена: 22000  Бренд: BMW, Страна: Германия, Год: 2018, Цена: 35000  Бренд: Toyota, Страна: Япония, Год: 2019, Цена: 25000 |

## 2.4 Удаление узла

Отобразим выполнение данной операции на рисунке 3.

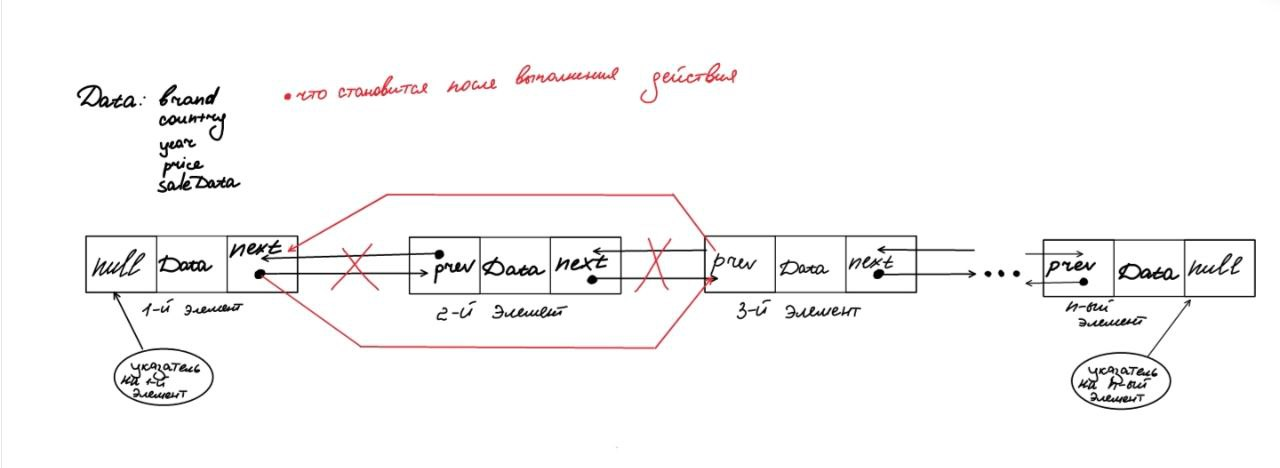


Рисунок 3 - Изображение удаления узла

Сначала метод проверяет, не является ли указатель node равным nullptr. Если это так, то ничего не делается, и метод завершается.

Затем метод проверяет, имеет ли удаляемый узел node предыдущий узел prev. Если у него есть предыдущий узел, то указатель next предыдущего узла обновляется так, чтобы он указывал на узел, следующий за удаляемым узлом. Если удаляемый узел был головным, то обновляется указатель head, чтобы он указывал на следующий узел после удаляемого. Затем метод проверяет, имеет ли удаляемый узел следующий узел. Если у него есть следующий узел, то указатель prev следующего узла обновляется так, чтобы он указывал на узел, предшествующий удаляемому узлу. Если удаляемый узел был хвостовым, то обновляется указатель tail, чтобы он указывал на предыдущий узел перед удаляемым. Удаляемый узел освобождается с помощью оператора delete.

Реализация данного алгоритма представлена в блок коде 3. Данные для тестирования будут приведены в таблице 2.

| // Удаление узла void List::deleteNode(Node\* node) {  if (node == nullptr)  return;   if (node->prev != nullptr)  node->prev->next = node->next;  else  head = node->next;   if (node->next != nullptr)  node->next->prev = node->prev;  else  tail = node->prev;   delete node; } |
| --- |

Листинг 3 - Удаление узла по ключу

Таблица 2 - Данные для тестирования

| № | Входные данные | Удалить | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Бренд: Audi, Страна: Германия, Год: 2021, Цена: 45000.0  Бренд: Ford, Страна: США, Год: 2016, Цена: 18000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Honda, Страна: Япония, Год: 2020, Цена: 22000  Бренд: BMW, Страна: Германия, Год: 2018, Цена: 35000  Бренд: Toyota, Страна: Япония, Год: 2019, Цена: 25000 | Audi, Германия, 2021, 45000.0 | Бренд: Ford, Страна: США, Год: 2016, Цена: 18000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Honda, Страна: Япония, Год: 2020, Цена: 22000  Бренд: BMW, Страна: Германия, Год: 2018, Цена: 35000  Бренд: Toyota, Страна: Япония, Год: 2019, Цена: 25000 |

## 2.5 Вывод элементов списка на экран.

Вывод элементов списка слева направо. Начинаем с головы списка и перебираем каждый элемент, выводя информацию о марке, стране, годе и цене автомобиля. Затем переходим к следующему элементу, используя указатель next, и повторяем процесс до тех пор, пока не достигнем конца списка.

Выводим элементы списка справа налево. Начинаем с хвоста списка и перебираем каждый элемент, также выводя информацию о марке, стране, годе и цене автомобиля. Затем переходим к предыдущему элементу, используя указатель prev, и повторяем процесс до тех пор, пока не достигнем начала списка. Реализация данного алгоритма представлена в блок коде 4.

| void List::printListForward() const {  Node\* current = head;  while (current != nullptr) {  cout << "Бренд: " << current->brand << ", Страна: " << current->country << ", Год: " << current->year << ", Цена: " << current->price << endl;  current = current->next;  } } void List::printListBackward() const {  Node\* current = tail;  while (current != nullptr) {  cout << "Бренд: " << current->brand << ", Страна: " << current->country << ", Год: " << current->year << ", Цена: " << current->price << endl;  current = current->prev;  } } |
| --- |

Листинг 4 - Вывод элементов на экран

## 2.6 Поиск узла с заданным значением.

Отобразим выполнение данной операции на рисунке 4.

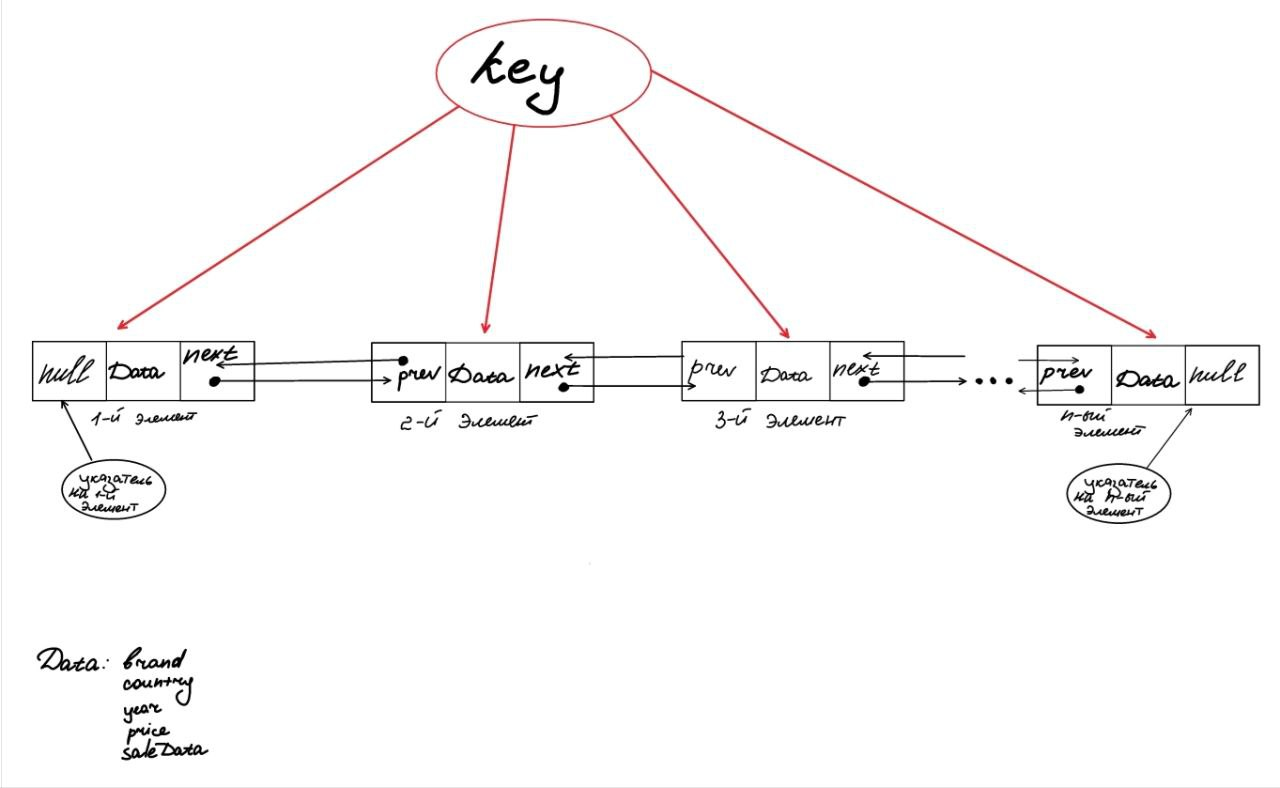


Рисунок 4 - Поиск узла с заданным значением

Метод начинает поиск с головного узла списка, инициализируя указатель current значением указателя head. Затем метод перебирает каждый узел списка, проверяя совпадение марки автомобиля текущего узла с заданной маркой (brand). Это осуществляется с помощью условного оператора if. Если марка автомобиля текущего узла соответствует заданной марке, метод возвращает указатель на текущий узел. Если узел с заданной маркой не найден во всем списке, метод завершается, возвращая nullptr. Этот алгоритм проходит по списку, находит самую длинную последовательность одинаковых символов и выводит информацию о ней.

Реализация данного алгоритма представлена в блок коде 5.

| // Поиск узла по марке автомобиля Node\* List::searchNode(string brand) const {  Node\* current = head;  while (current != nullptr) {  if (current->brand == brand)  return current;  current = current->next;  }  return nullptr; // Узел с заданной маркой не найден } |
| --- |

Листинг 5 - Поиск узла с заданным значением

## 2.7 Упорядочить созданный список из n узлов по стране изготовителю.

Метод сортирует список узлов по стране изготовителю с использованием сортировки вставками. Если список пуст или содержит один элемент, он уже упорядочен. В противном случае, создается новый упорядоченный список, в который поочередно вставляются узлы исходного списка в правильные позиции. Указатели обновляются соответственно, и по завершении сортировки обновляется указатель head, чтобы он указывал на головной элемент нового упорядоченного списка.

Оценим сложность данного алгоритма. Алгоритм упорядочивания списка по стране изготовителю, реализованный в функции orderListByCountry(), использует сортировку вставками. В общем случае, если список содержит n элементов, сложность данного алгоритма будет O(n2).

Реализация данного алгоритма представлена в блок коде 6. Данные для тестирования будут приведены в таблице 3.

| // Упорядочить список по стране изготовителю void List::orderListByCountry() {  if (head == nullptr || head->next == nullptr)  return; // Если список пуст или содержит только один элемент, то он уже упорядочен по стране   // Применение сортировки вставками для упорядочивания списка по стране изготовителю  Node\* sorted = nullptr; // Инициализация нового списка, который будет упорядоченным по стране  Node\* current = head;   while (current != nullptr) {  Node\* next = current->next;  if (sorted == nullptr || current->country < sorted->country) {  // Вставляем узел в начало отсортированного списка  current->prev = nullptr;  current->next = sorted;  if (sorted != nullptr)  sorted->prev = current;  sorted = current;  }  else {  // Ищем правильную позицию в отсортированном списке  Node\* search = sorted;  while (search->next != nullptr && search->next->country < current->country)  search = search->next;  current->next = search->next;  current->prev = search;  if (search->next != nullptr)  search->next->prev = current;  search->next = current;  }  current = next;  }   // Обновляем голову списка  head = sorted; } |
| --- |

Листинг 6 - Упорядочить созданный список из n узлов по стране изготовителю

Таблица 3 - Данные для тестирования

| № | Входные данные | Ожидаемый результат |
| --- | --- | --- |
| 1 | Бренд: Ford, Страна: США, Год: 2016, Цена: 18000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Honda, Страна: Япония, Год: 2020, Цена: 22000  Бренд: BMW, Страна: Германия, Год: 2018, Цена: 35000  Бренд: Toyota, Страна: Япония, Год: 2019, Цена: 25000 | Бренд: Audi, Страна: Германия, Год: 2021, Цена: 45000  Бренд: BMW, Страна: Германия, Год: 2018, Цена: 35000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Ford, Страна: США, Год: 2016, Цена: 18000  Бренд: Honda, Страна: Япония, Год: 2020, Цена: 22000  Бренд: Toyota, Страна: Япония, Год: 2019, Цена: 25000 |

## 2.8 Установить дату продажи автомобиля.

Примем два параметра: указатель на узел node, для которого нужно установить дату продажи, и строку date, содержащую дату продажи. Сначала метод проверяет, не является ли указатель node равным nullptr. Если указатель не равен nullptr, это означает, что узел существует в списке, и можно установить для него дату продажи. В противном случае метод завершается, не выполняя никаких действий. Если указатель node не равен nullptr, метод присваивает строку date полю Date узла node. Таким образом, дата продажи успешно устанавливается для заданного узла в списке.

Реализация данного алгоритма представлена в блок коде 7. Данные для тестирования будут приведены в таблице 4.

| // Установить дату продажи void List::setDate(Node\* node, string date) {  if (node != nullptr)  node->Date = date; } |
| --- |

Листинг 7 - Установить дату продажи проданному автомобилю

## 2.9 Удалить все узлы по проданным автомобилям.

Начнем обход списка, инициализируя указатель current значением указателя head. Для каждого узла в списке проверяется, установлена ли у него дата продажи (Date). Это делается с помощью метода empty(), который возвращает true, если строка пуста, и false в противном случае. Если у текущего узла установлена дата продажи (строка Date не пуста), то сохраняется указатель на следующий узел (next), чтобы предотвратить ошибку при удалении текущего узла; вызывается метод deleteNode, который удаляет текущий узел из списка. Указатель current обновляется, чтобы указывать на следующий узел после удаления текущего узла. Если у текущего узла дата продажи не установлена (строка Date пуста), то просто переходим к следующему узлу. Процесс продолжается до тех пор, пока не будут проверены все узлы в списке.

Реализация данного алгоритма представлена в блок коде 8. Данные для тестирования будут приведены в таблице 4.

| // Удалить все узлы по проданным автомобилям void List::deleteSoldCars() {  Node\* current = head;  while (current != nullptr) {  if (!current->Date.empty()) {  Node\* next = current->next;  deleteNode(current);  current = next;  }  else {  current = current->next;  }  } } |
| --- |

Листинг 8 - Удалить все узлы по проданным автомобилям

Таблица 4 - Данные для тестирования

| № | Входные данные | Дата | Ожидаемый результат |
| --- | --- | --- | --- |
| 1 | Бренд: Audi, Страна: Германия, Год: 2021, Цена: 45000  Бренд: BMW, Страна: Германия, Год: 2018, Цена: 35000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Ford, Страна: США, Год: 2016, Цена: 18000  Бренд: Honda, Страна: Япония, Год: 2020, Цена: 22000  Бренд: Toyota, Страна: Япония, Год: 2019, Цена: 25000 | 2024-04-23 | Найденный автомобиль: Honda Япония 2020 22000  Список после удаления проданных автомобилей:  Бренд: Audi, Страна: Германия, Год: 2021, Цена: 45000  Бренд: BMW, Страна: Германия, Год: 2018, Цена: 35000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Ford, Страна: США, Год: 2016, Цена: 18000  Бренд: Toyota, Страна: Япония, Год: 2019, Цена: 25000 |

## **2.**10 **Реализация алгоритма на языке C++ и проведение тестирования**

### **2.10.1 Реализация на языке программирования C++**

Реализуем данный алгоритм на языке C++. Для реализации понадобятся такие библиотеки, как iostream, string.

| #ifndef LIST\_H #define LIST\_H  #include <string> #include <iostream> using namespace std;  struct Node {  string brand;  string country;  int year;  double price;  string Date;  Node\* prev;  Node\* next; };  class List { public:  List(); // конструктор  ~List(); // деструктор   void deleteNode(Node\* node); // удаление узла  void printListForward() const; // вывод списка слева направо  void printListBackward() const; // вывод списка справа налево  Node\* searchNode(string brand) const; // поиск узла по марке автомобиля   // Дополнительные операции  void orderListByCountry(); // упорядочить список по стране изготовителю  void insertAtBeginning(string brand, string country, int year, double price); // вставить узел в начало подсписка  void setDate(Node\* node, string date); // установить дату продажи  void deleteSoldCars(); // удалить все узлы по проданным автомобилям  private:  Node\* head;  Node\* tail; };  #endif |
| --- |

Листинг 9 – list.h

| #include "list.h"  // Конструктор List::List() {  head = nullptr;  tail = nullptr; }  // Деструктор List::~List() {  Node\* current = head;  while (current != nullptr) {  Node\* next = current->next;  delete current;  current = next;  } } // Удаление узла void List::deleteNode(Node\* node) {  if (node == nullptr)  return;   if (node->prev != nullptr)  node->prev->next = node->next;  else  head = node->next;   if (node->next != nullptr)  node->next->prev = node->prev;  else  tail = node->prev;   delete node; }  // Вывод списка слева направо void List::printListForward() const {  Node\* current = head;  while (current != nullptr) {  cout << "Бренд: " << current->brand << ", Страна: " << current->country << ", Год: " << current->year << ", Цена: " << current->price << endl;  current = current->next;  } }  // Вывод списка справа налево void List::printListBackward() const {  Node\* current = tail;  while (current != nullptr) {  cout << "Бренд: " << current->brand << ", Страна: " << current->country << ", Год: " << current->year << ", Цена: " << current->price << endl;  current = current->prev;  } }  // Поиск узла по марке автомобиля Node\* List::searchNode(string brand) const {  Node\* current = head;  while (current != nullptr) {  if (current->brand == brand)  return current;  current = current->next;  }  return nullptr; // Узел с заданной маркой не найден }  // Дополнительные операции // Вставить узел в начало подсписка void List::insertAtBeginning(string brand, string country, int year, double price) {  Node\* newNode = new Node;  newNode->brand = brand;  newNode->country = country;  newNode->year = year;  newNode->price = price;  newNode->Date = ""; // Пока не продан   // Если список пуст, вставляем узел как головной  if (head == nullptr) {  newNode->prev = nullptr;  newNode->next = nullptr;  head = newNode;  tail = newNode;  }  else {  // Находим первый узел с той же страной изготовителем и вставляем перед ним  Node\* current = head;  while (current != nullptr && current->country == country)  current = current->next;   if (current == head) {  // Вставляем перед текущим головным узлом  newNode->prev = nullptr;  newNode->next = head;  head->prev = newNode;  head = newNode;  }  else if (current == nullptr) {  // Если такой страны нет в списке, вставляем в конец  newNode->prev = tail;  newNode->next = nullptr;  tail->next = newNode;  tail = newNode;  }  else {  // Вставляем перед узлом с той же страной изготовителем  newNode->prev = current->prev;  newNode->next = current;  current->prev->next = newNode;  current->prev = newNode;  }  } }   // Установить дату продажи void List::setDate(Node\* node, string date) {  if (node != nullptr)  node->Date = date; }  // Удалить все узлы по проданным автомобилям void List::deleteSoldCars() {  Node\* current = head;  while (current != nullptr) {  if (!current->Date.empty()) {  Node\* next = current->next;  deleteNode(current);  current = next;  }  else {  current = current->next;  }  } }  // Упорядочить список по стране изготовителю void List::orderListByCountry() {  if (head == nullptr || head->next == nullptr)  return; // Если список пуст или содержит только один элемент, то он уже упорядочен по стране   // Применение сортировки вставками для упорядочивания списка по стране изготовителю  Node\* sorted = nullptr; // Инициализация нового списка, который будет упорядоченным по стране  Node\* current = head;   while (current != nullptr) {  Node\* next = current->next;  if (sorted == nullptr || current->country < sorted->country) {  // Вставляем узел в начало отсортированного списка  current->prev = nullptr;  current->next = sorted;  if (sorted != nullptr)  sorted->prev = current;  sorted = current;  }  else {  // Ищем правильную позицию в отсортированном списке  Node\* search = sorted;  while (search->next != nullptr && search->next->country < current->country)  search = search->next;  current->next = search->next;  current->prev = search;  if (search->next != nullptr)  search->next->prev = current;  search->next = current;  }  current = next;  }   // Обновляем голову списка  head = sorted; } |
| --- |

Листинг 10 – list.cpp

| #include <iostream> #include "list.h" using namespace std;  int main() {  setlocale(LC\_ALL, "RUS");  List List;  string brand, country, Date;  int year;  double price;  Node\* nodeToDelete = nullptr;  Node\* searchResult = nullptr;  // Создание списка  List.insertAtBeginning("Toyota", "Япония", 2019, 25000.0);  List.insertAtBeginning("BMW", "Германия", 2018, 35000.0);  List.insertAtBeginning("Honda", "Япония", 2020, 22000.0);  List.insertAtBeginning("Mercedes", "Германия", 2017, 40000.0);  List.insertAtBeginning("Ford", "США", 2016, 18000.0);   cout << "Исходный список:" << endl;  List.printListForward();  cout << endl;  int choice\_method;   do {  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  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 начало подсписка" << endl;  cout << "4. Удаление узла" << endl;  cout << "5. Упорядочивание списка по стране автомобиля" << endl;  cout << "6. Поиск автомобиля по бренду, дата продажи и удаление проданных машин из списка" << endl;  cout << "7. Выход из программы" << endl;  cout << "\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_" << endl;  cout << "Введите пункт от 1 до 7: ";   cin >> choice\_method;   switch (choice\_method) {   case 1:  cout << "Список вперед:" << endl;  List.printListForward();  cout << endl;  break;   case 2:  cout << "Список назад:" << endl;  List.printListBackward();  cout << endl;  break;   case 3:  // Вставка нового узла в начало подсписка  List.insertAtBeginning("Audi", "Германия", 2021, 45000.0);  cout << "Список после добавления Audi в начало списка немецких автомобилей:" << endl;  List.printListForward();  cout << endl;  break;   case 4:  nodeToDelete = List.searchNode("Audi");  if (nodeToDelete != nullptr) {  List.deleteNode(nodeToDelete);  }  else {  cout << "Узел с указанными данными не найден." << endl;  }  break;   case 5:  // Упорядочивание списка по стране изготовителю  List.orderListByCountry();  cout << "Список после сортировки по стране:" << endl;  List.printListForward();  cout << endl;  break;   case 6:  searchResult = List.searchNode("Honda");  if (searchResult != nullptr) {  cout << "Найденный автомобиль: " << searchResult->brand << " " << searchResult->country << " " << searchResult->year << " " << searchResult->price << endl;  }  else {  cout << "Автомобиль не найден." << endl;  }   // Установка даты продажи  List.setDate(searchResult, "2024-04-23");   // Удаление проданных автомобилей  List.deleteSoldCars();  cout << "Список после удаления проданных автомобилей:" << endl;  List.printListForward();  break;   case 7:  cout << "Завершение программы" << endl;  return 0;   default:  cout << "Ошибка" << endl;  return 0;  }   } while (choice\_method != 7);   return 0; } |
| --- |

Листинг 11 – main.cpp

### **2.10.2 Тестирование**

Проведем тестирование основываясь на данные из таблиц 1-4. Результаты представлены на рисунках 5-9.

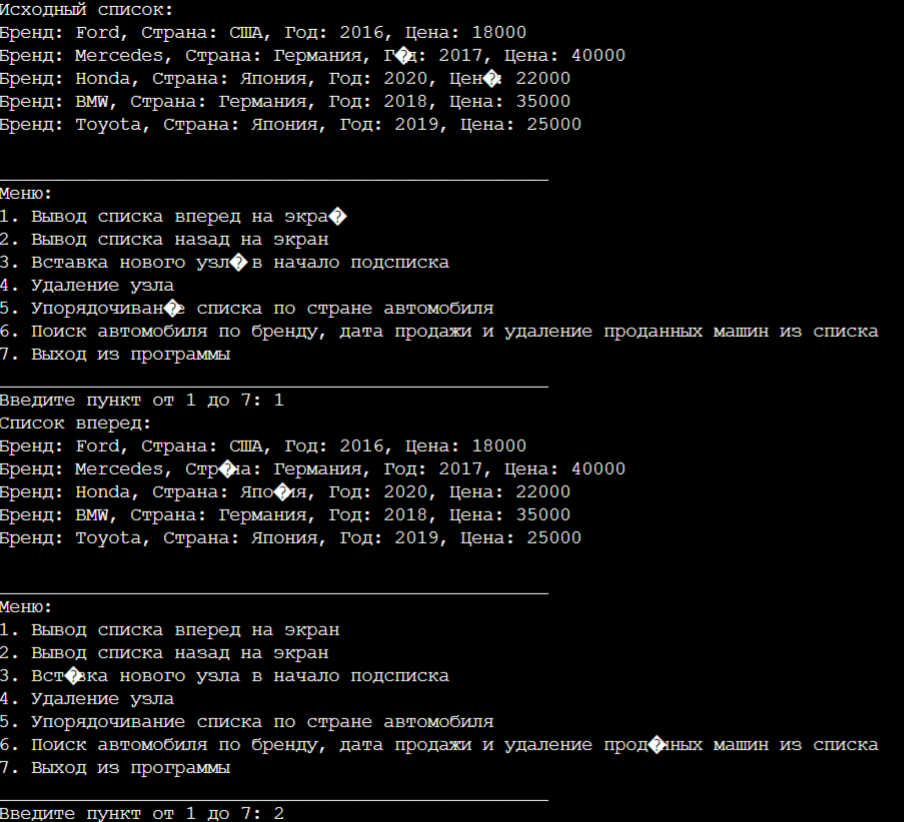


Рисунок 5 - Тестирование программы

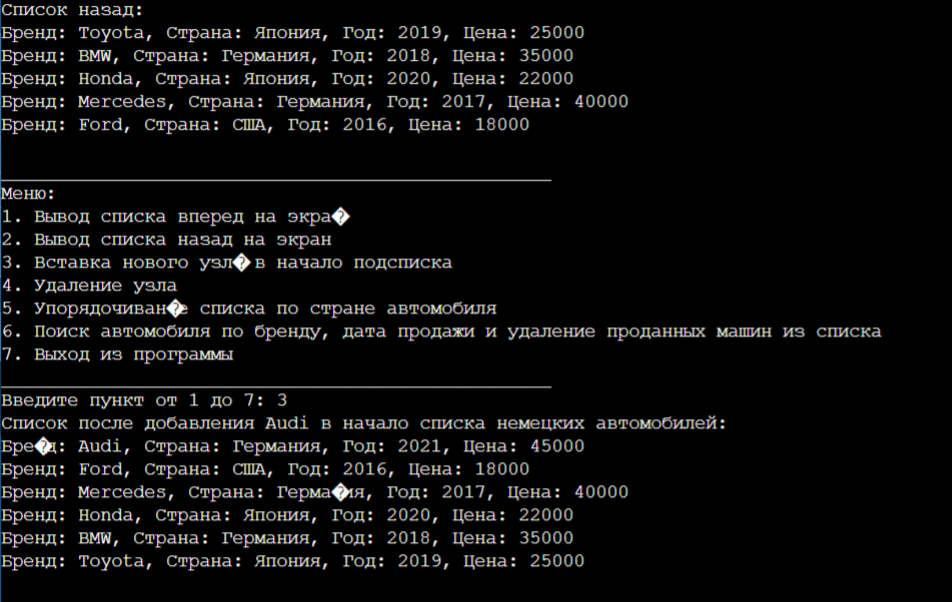


Рисунок 6 - Тестирование программы

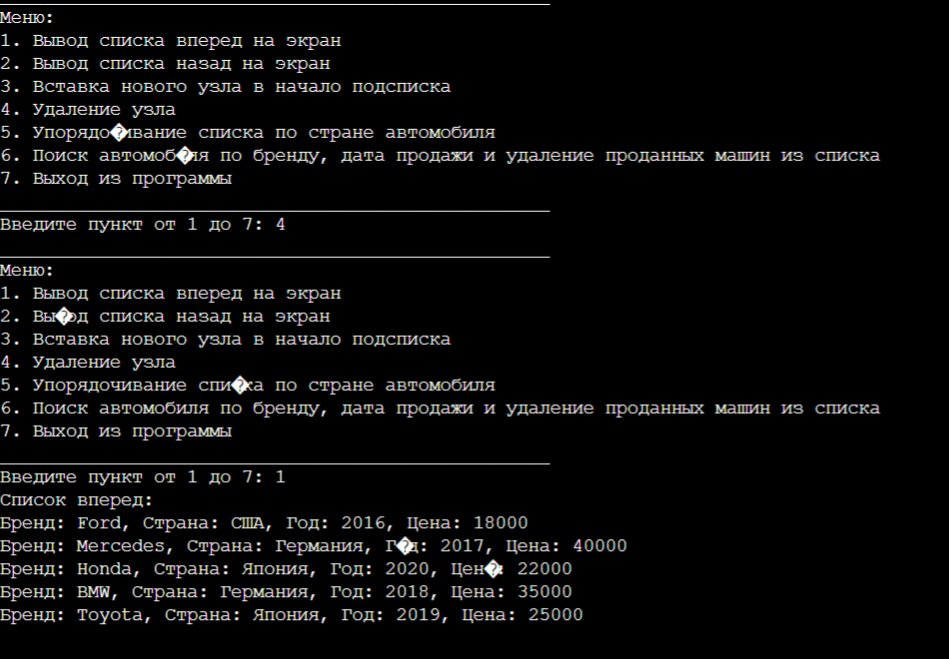


Рисунок 7 - Тестирование программы

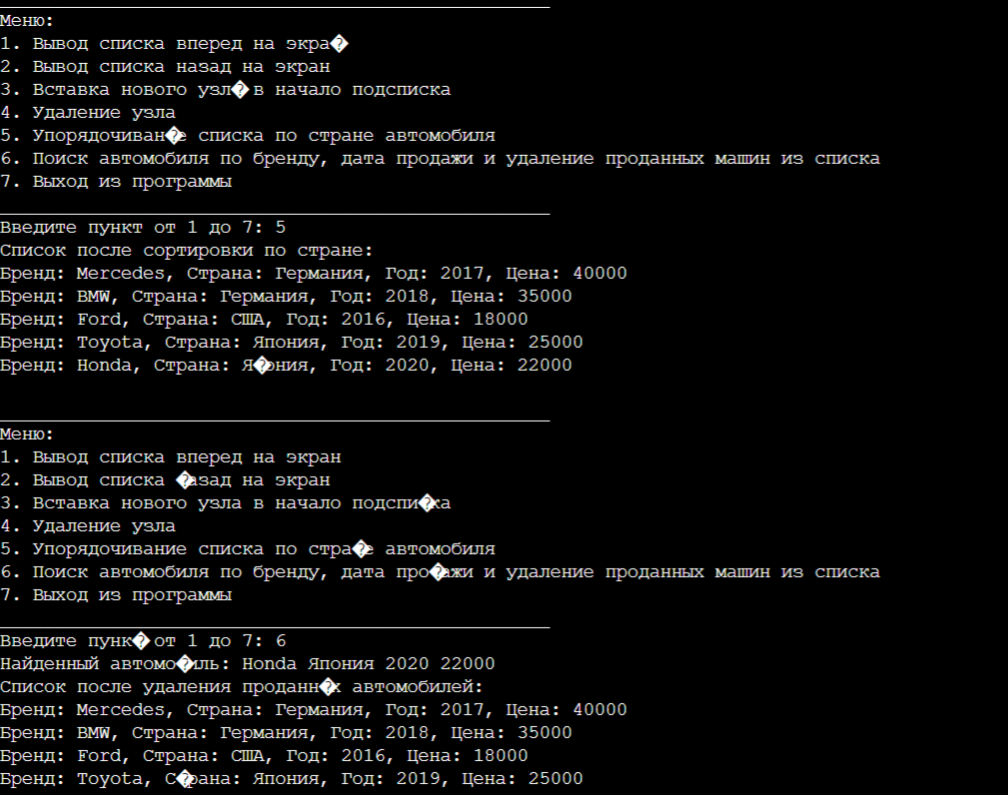


Рисунок 8 - Тестирование программы

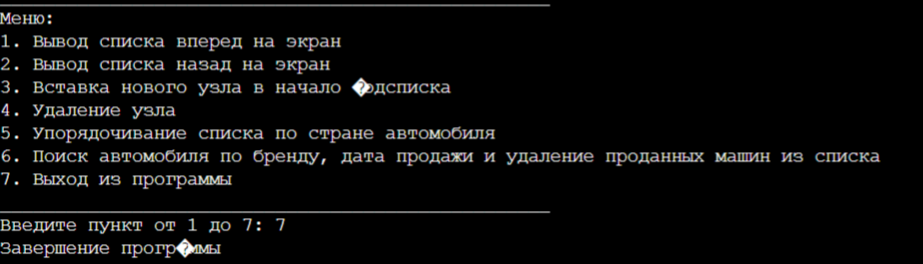


Рисунок 9 - Тестирование программы

Результаты тестирования совпали с ожидаемым результатом. Следовательно работа выполнена успешно.

# 

# **3 ВЫВОДЫ**

В ходе практической работы были выполнены следующие задачи:

- Получены знания по управления двунаправленным списком в программах на языке С++;

- Получены практические навыки управления двунаправленным списком в программах на языке С++;

- Проведён анализ структуры в соответствии с индивидуальным вариантом;

- Проведён анализ операций в соответствии с индивидуальным вариантом;

- Была реализована программа для структуры в соответствии с индивидуальным вариантом;

- Были реализованы программы для операций в соответствии с индивидуальным вариантом;

- Было реализовано меню выбора способа ввода и реализации функций;

- Было оценена сложность первой дополнительной операции;

- Проведено тестирование программы с различными данными и способами ввода;

Таким образом, главную цель практической работы, а именно получение знаний и практических навыков управления двунаправленным списком в программах на языке С++, можно считать выполненной.

# **4 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).