|  |
| --- |
| 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: Харченко А.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 списка и описание операций над списком 5](#_3znysh7)

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

[2.2.2 Процесс выполнения операций 6](#_tyjcwt)

[2.2.2.1 Вставка узла. 6](#_rjgck4c3lvht)

[2.2.2.2 Удаление узла. 7](#_iey0ptwr19lr)

[2.2.2.3 Вывод элементов списка на экран. 8](#_8z2ypeeoe7oc)

[2.2.2.4 Поиск узла с заданным значением. 9](#_65gkb5tnjygt)

[2.2.2.5 Упорядочить созданный список из n узлов по стране изготовителю. 10](#_4wy3p1rbi9su)

[2.2.2.6 Вставка узла в начале списка. 10](#_jeetx1bwvhy7)

[2.2.2.7 Установить дату продажи автомобиля. 11](#_9ix0efjfz3x4)

[2.2.2.8 Удалить все узлы по проданным автомобилям. 11](#_jx91y1l5b9p0)

[2.3 Реализация алгоритма на языке C++ и проведение тестирования 12](#_3dy6vkm)

[2.3.1 Реализация на языке программирования C++ 12](#_1t3h5sf)

[2.3.2 Тестирование 20](#_4d34og8)

[2.4 Вывод по заданию 23](#_2s8eyo1)

[3 ВЫВОДЫ 24](#_lnxbz9)

[4 ЛИТЕРАТУРА 25](#_35nkun2)

# **1 ЦЕЛЬ**

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

# **2 ЗАДАНИЕ**

## **2.1 Формулировка задачи**

Вариант 10, в списке 27.

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

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.

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 списка и описание операций над списком**

### **2.2.1 Определение структуры узла двунаправленного списка**

Эта структура представляет собой узел двунаправленного списка машин. Каждый узел содержит следующие поля:

brand: марка машины (тип string),

country: страна производства машины (тип string),

year: год выпуска машины (тип int),

price: цена машины (тип double),

Date: дата продажи машины (тип string),

prev: указатель на предыдущий узел в списке (тип Node\*),

next: указатель на следующий узел в списке (тип Node\*).

Представим структуру в виде кода в блоке кода 1 и изобразим на рисунке 1.

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

Блок кода 1 - Структура

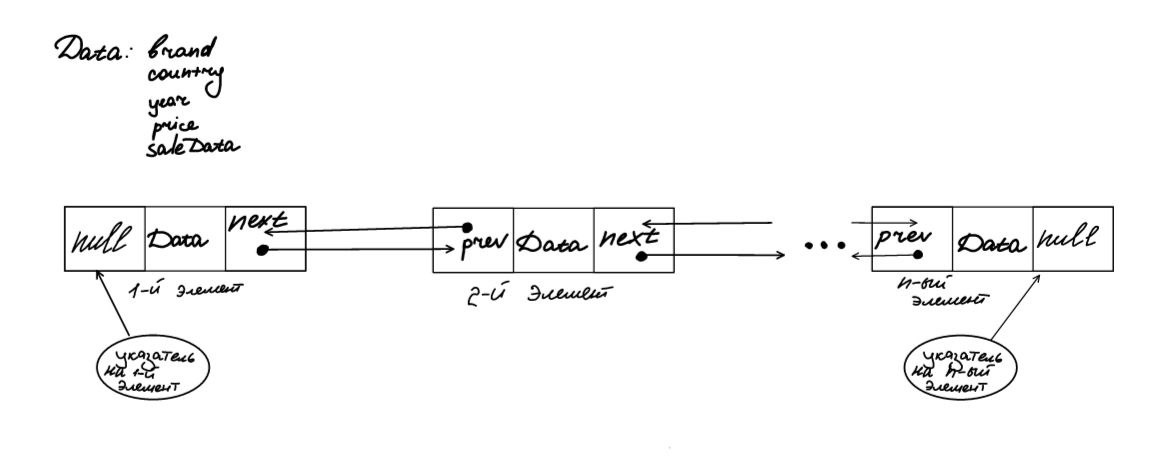


Рисунок 1 - Изображение структуры данных

### **2.2.2 Процесс выполнения операций**

На основании индивидуального варианта, можно сказать, что используются операции: 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), упорядочить созданный список из n узлов так, чтобы узлы были упорядочены по стране изготовителю (будут сформированы подсписки по стране), установить дату продажи проданному автомобилю, удалить все узлы по проданным автомобилям, Вставка новый узла в начало подсписка.

Изобразим, рассмотрим алгоритм, реализуем и предоставим таблицу с данными для тестирования данных операций.

#### 2.2.2.1 Вставка узла.

Отобразим выполнение данной операции на рисунке 2.

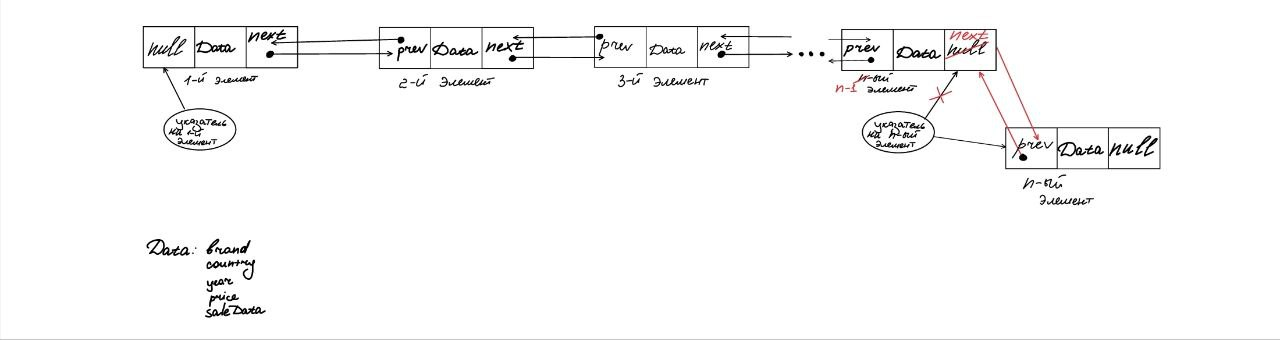


Рисунок 2 - Изображение добавление нового узла

Добавим новый узел в конец двунаправленного списка машин. Узел содержит информацию о машине, такую как марка, страна производства, год выпуска, цена и дата продажи. Если список пуст, новый узел становится головой и хвостом списка, иначе он добавляется за последним узлом.

Реализация данного алгоритма представлена в блок коде 3. Данные для тестирования будут приведены в таблице 1.

Таблица 1 - Данные для тестирования

| Входные данные | Добавить | Ожидаемый результат |
| --- | --- | --- |
| Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000 | Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000 | Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000 |

#### 2.2.2.2 Удаление узла.

Отобразим выполнение данной операции на рисунке 3.

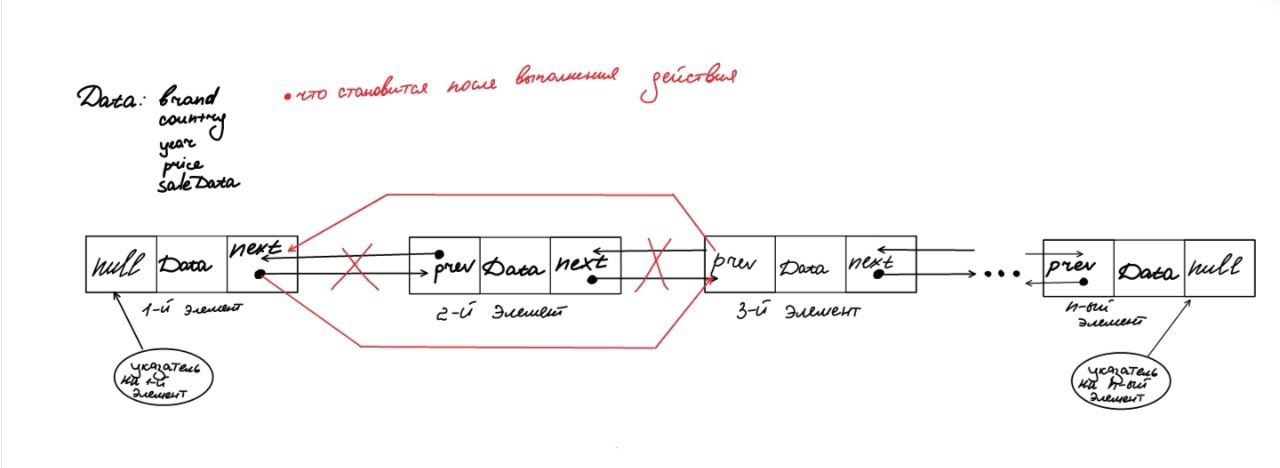


Рисунок 3 - Изображение удаления узла

Удалим указанный узел из двунаправленного списка машин. Проверяется, существует ли узел для удаления. Если он не равен nullptr, то обновляются ссылки на предыдущий и следующий узлы, чтобы пропустить удаляемый узел. Затем удаляемый узел освобождается из памяти.

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

Таблица 2 - Данные для тестирования

| Входные данные | Удалить | Ожидаемый результат |
| --- | --- | --- |
| Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000 | Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000 | Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000 |

#### 2.2.2.3 Вывод элементов списка на экран.

Вывод элементов списка на экран может быть осуществлен в двух направлениях (вперед и назад). Рассмотрим оба направления.

Начиная с головы списка (head), метод перебирает все узлы по порядку, печатая информацию о каждой машине, включая бренд, страну, год выпуска и цену. После печати информации текущий указатель переключается на следующий узел, пока не будет достигнут конец списка (указатель nullptr).

Начиная с хвоста списка (tail), метод перебирает все узлы в обратном порядке, печатая информацию о каждой машине, включая бренд, страну, год выпуска и цену. После печати информации текущий указатель переключается на предыдущий узел, пока не будет достигнут начало списка (указатель nullptr).

Реализация данного алгоритма представлена в блок коде 3. Данные для тестирования слева направо будут приведены в таблице 3, а для справа налево в таблице 4.

Таблица 3 - Данные для тестирования

| Входные данные | Ожидаемый результат |
| --- | --- |
| Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000 | Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000 |

Таблица 4 - Данные для тестирования

| Входные данные | Ожидаемый результат |
| --- | --- |
| Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000 | Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000 |

#### 2.2.2.4 Поиск узла с заданным значением.

Начиная с головы списка, метод перебирает узлы по порядку. При нахождении узла с брендом, совпадающим с заданным, возвращается указатель на этот узел. Если узел с заданным брендом не найден, возвращается nullptr.

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

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

Если список пуст или содержит только один элемент, метод завершается. Создается новый список sorted, который будет отсортированным. Начиная с головы списка, каждый узел перемещается из исходного списка в отсортированный таким образом, чтобы список оставался упорядоченным по стране производства. Узлы перемещаются сравнением страны текущего узла с странами узлов в отсортированной части списка. После завершения сортировки, указатель head обновляется на указатель на начало отсортированного списка.

Общее время выполнения составляет O(n2), что делает этот алгоритм неэффективным для больших списков. Если список уже отсортирован, сложность может снизиться до O(n), но в худшем случае он все равно будет O(n2).

Реализация данного алгоритма представлена в блок коде 3. Данные для тестирования будут приведены в таблице 5.

Таблица 5 - Данные для тестирования

| Входные данные | Ожидаемый результат |
| --- | --- |
|  | Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000 |

#### 2.2.2.6 Вставка узла в начале списка.

Добавляет новый узел в начало списка, принимая в качестве параметров марку, страну, год выпуска и цену машины. Если список пуст, новый узел становится единственным узлом и как головой, так и хвостом списка. Если список не пуст, новый узел вставляется перед первым узлом, если машина с такой же страной производства уже существует, иначе он вставляется в конец списка.

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

Таблица 6 - Данные для тестирования

| Входные данные | Добавить | Ожидаемый результат |
| --- | --- | --- |
| Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000 | Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000 | Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000 |

#### 2.2.2.7 Установить дату продажи автомобиля.

Метод принимает два параметра: указатель на узел node, для которого нужно установить дату продажи, и строку date, содержащую дату продажи. Установим дату продажи для указанного узла.

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

#### 2.2.2.8 Удалить все узлы по проданным автомобилям.

Удаляет из списка все проданные машины, перемещаясь по списку и удаляя узлы, у которых установлена дата продажи.

Реализация данного алгоритма представлена в блок коде 3. Данные для тестирования будут приведены в таблице 7.

Таблица 7 - Данные для тестирования

| Входные данные | Дата | Ожидаемый результат |
| --- | --- | --- |
| Бренд: Audi, Страна: Германия, Год: 2015, Цена: 30000  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000 | 22-03-2005 | Введите бренд для поиска: Audi  Найденный автомобиль: Audi Германия 2015 30000  Введите дату продажи автомобиля: 22-03-2005  Список после удаления проданных автомобилей:  Бренд: Mercedes, Страна: Германия, Год: 2017, Цена: 40000  Бренд: Volkswagen, Страна: Германия, Год: 2019, Цена: 28000  Бренд: Chevrolet, Страна: США, Год: 2017, Цена: 20000  Бренд: Hyundai, Страна: Южная Корея, Год: 2019, Цена: 21000  Бренд: Nissan, Страна: Япония, Год: 2018, Цена: 23000 |

## **2.3 Реализация алгоритма на языке C++ и проведение тестирования**

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

Реализуем данный алгоритм на языке C++(блок схема 2-4). Для реализации понадобятся такие библиотеки, как iostream, string.

| #ifndef CAR\_H #define CAR\_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 Car { public:  Car();  ~Car();   void insertNode(string brand, string country, int year, double price); // вставка узла  void deleteNode(Node\* node);  void printListForward() const;  void printListBackward() const;  Node\* searchNode(string brand) const;  void orderListByCountry();  void insertNodeAtBeginning(string brand, string country, int year, double price);  void setDate(Node\* node, string date);  void deleteSoldCars();  private:  Node\* head;  Node\* tail; };  #endif |
| --- |

Блок кода 2 – car.h

| #include "car.h"  Car::Car() {  head = nullptr;  tail = nullptr; }  Car::~Car() {  Node\* current = head;  while (current != nullptr) {  Node\* next = current->next;  delete current;  current = next;  } }  void Car::insertNode(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  newNode->prev = nullptr;  newNode->next = nullptr;   if (head == nullptr) {  head = newNode;  tail = newNode;  }  else {  tail->next = newNode;  newNode->prev = tail;  tail = newNode;  } }  void Car::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; }  void Car::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 Car::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;  } }  Node\* Car::searchNode(string brand) const {  Node\* current = head;  while (current != nullptr) {  if (current->brand == brand)  return current;  current = current->next;  }  return nullptr; }  void Car::insertNodeAtBeginning(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 = "";   if (head == nullptr) {  newNode->prev = nullptr;  newNode->next = nullptr;  head = newNode;  tail = newNode;  }  else {  Node\* current = head;  while (current != nullptr && current->country == country)  current = current->next;   if (current == head) {  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 {  newNode->prev = current->prev;  newNode->next = current;  current->prev->next = newNode;  current->prev = newNode;  }  } }  void Car::setDate(Node\* node, string date) {  if (node != nullptr)  node->Date = date; }  void Car::deleteSoldCars() {  Node\* current = head;  while (current != nullptr) {  if (!current->Date.empty()) {  Node\* next = current->next;  deleteNode(current);  current = next;  }  else {  current = current->next;  }  } }  void Car::orderListByCountry() {  if (head == nullptr || head->next == nullptr)  return;  Node\* sorted = nullptr;  Node\* current = head;  while (current != nullptr) {  Node\* next = current->next;  if (sorted == nullptr || current->country < sorted->country) {  current->prev = nullptr;  current->next = sorted;  if (sorted != nullptr)  sorted->prev = current;  sorted = current;  }  else {  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;  }  head = sorted; } |
| --- |

Блок кода 3 – car.cpp

| #include <iostream> #include "car.h" using namespace std;  int main() {  setlocale(LC\_ALL, "RUS");  Car car;  string brand, country, Date;  int year;  double price;  car.insertNode("Audi", "Германия", 2015, 30000.0);  car.insertNode("Chevrolet", "США", 2017, 20000.0);  car.insertNode("Volkswagen", "Германия", 2019, 28000.0);  car.insertNode("Nissan", "Япония", 2018, 23000.0);  car.insertNode("Hyundai", "Южная Корея", 2019, 21000.0);  cout << "Исходный список:" << endl;  car.printListForward();  cout << endl;  int choice\_method;  Node \* nodeToDelete; // Declare outside the do-while loop   do {  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. Вставка нового узла в начало подсписка" << endl;  cout << "5. Удаление узла" << endl;  cout << "6. Упорядочивание списка по стране автомобиля" << endl;  cout << "7. Поиск автомобиля по бренду, дата продажи и удаление проданных машин из списка" << endl;  cout << "8. Выход из программы" << endl;  cout << "Введите пункт от 1 до 8: ";   cin >> choice\_method;   switch (choice\_method) {   case 1:  cout << "Список вперед:" << endl;  car.printListForward();  cout << endl;  break;   case 2:  cout << "Список назад:" << endl;  car.printListBackward();  cout << endl;  break;   case 3:  car.insertNode("Mercedes", "Германия", 2017, 40000.0);  cout << "Список после добавления Audi в конец списка немецких автомобилей:" << endl;  car.printListForward();  cout << endl;  break;   case 4:  car.insertNodeAtBeginning("Mercedes", "Германия", 2017, 40000.0);  cout << "Список после добавления Mercedes в начало списка немецких автомобилей:" << endl;  car.printListForward();  cout << endl;  break;  case 5:  nodeToDelete = car.searchNode("Mercedes"); // Assign value here  if (nodeToDelete != nullptr) {  car.deleteNode(nodeToDelete);  } else {  cout << "Узел с указанными данными не найден." << endl;  }  break;  case 6:  car.orderListByCountry();  cout << "Список после упорядочивания по стране:" << endl;  car.printListForward();  cout << endl;  break;  case 7:  cout << "Введите бренд для поиска: ";  cin >> brand;  {  Node\* searchResult = car.searchNode(brand);  if (searchResult != nullptr) {  cout << "Найденный автомобиль: " << searchResult->brand << " " << searchResult->country << " " << searchResult->year << " " << searchResult->price << endl;  }   else {  cout << "Автомобиль не найден." << endl;  }  if (searchResult != nullptr) {  cout << "Введите дату продажи автомобиля: ";  cin >> Date;  car.setDate(searchResult, Date);  }  }  car.deleteSoldCars();  cout << "Список после удаления проданных автомобилей:" << endl;  car.printListForward();  break;  case 8:  cout << "Завершение программы" << endl;  return 0;   default:  cout << "Ошибка" << endl;  return 0;  }   } while (choice\_method != 8);   return 0; } |
| --- |

Блок кода 4 – main.cpp

### **2.3.2 Тестирование**

Проведем тестирование основываясь на данные из таблиц 1-7. Результаты тестирования представлены на рисунках 4-7.

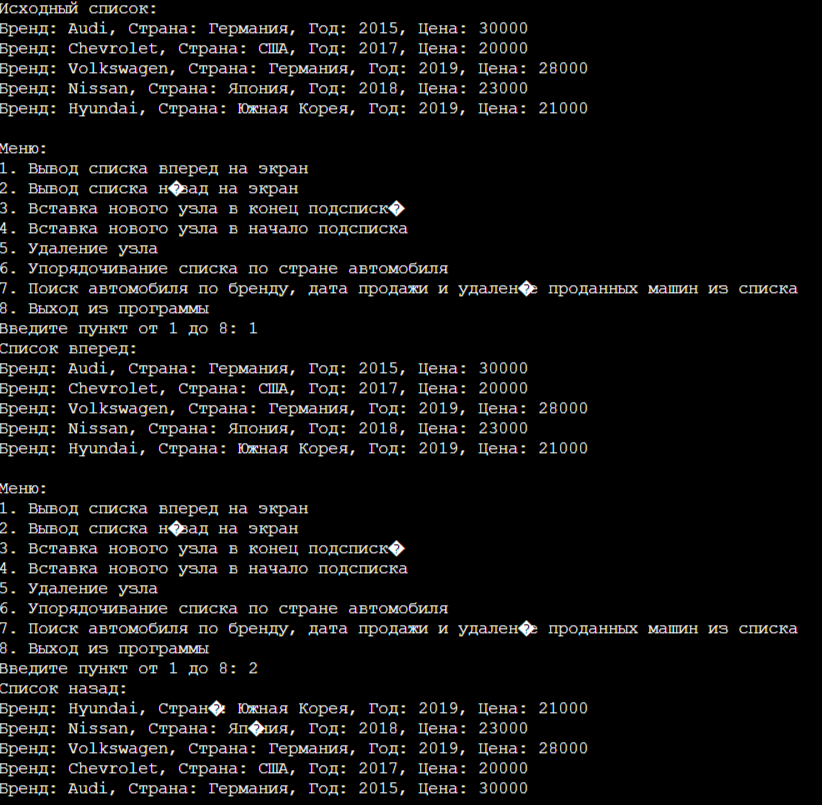


Рисунок 4 - Тестирование программы

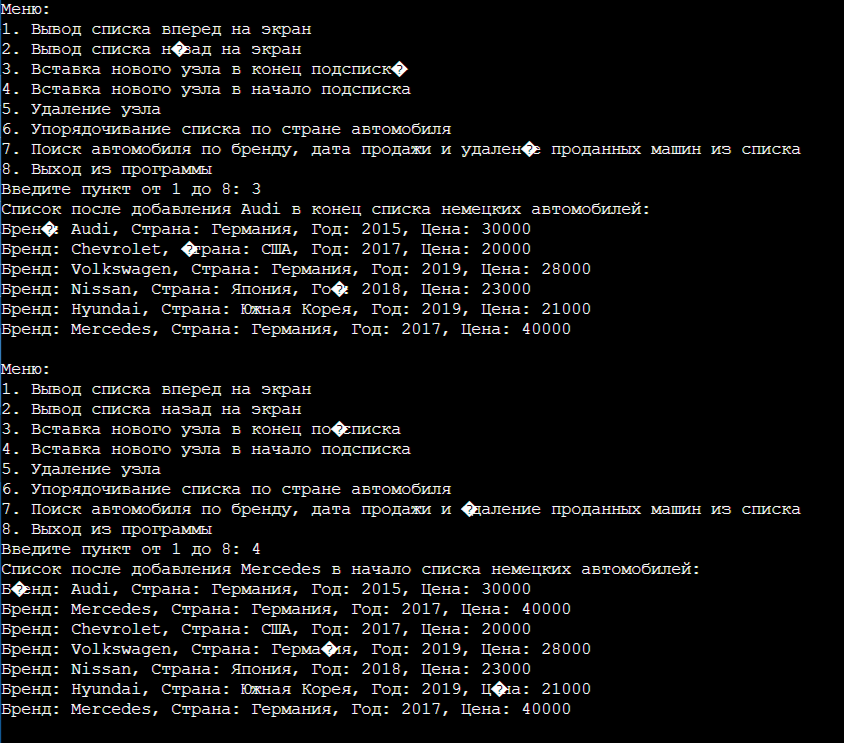


Рисунок 5 - Тестирование программы

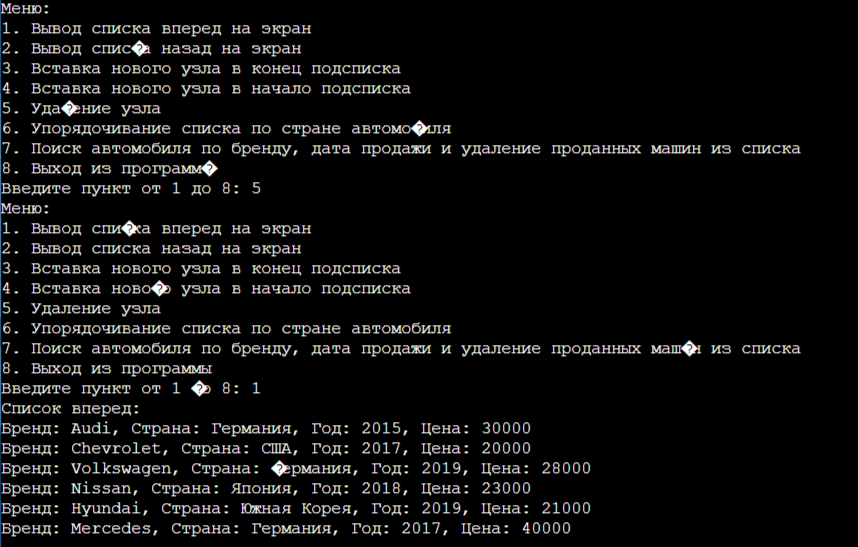


Рисунок 6 - Тестирование программы

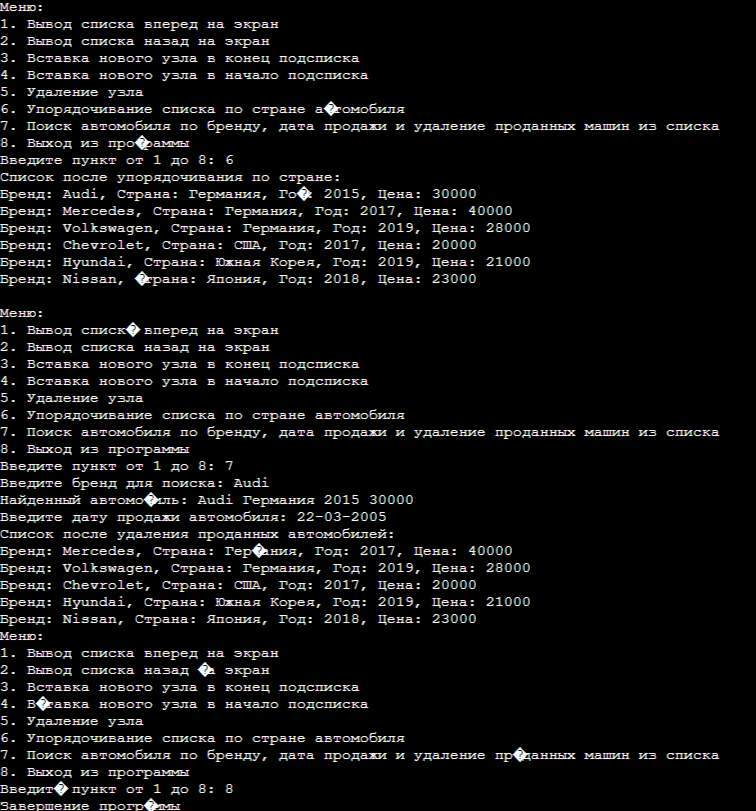


Рисунок 7 - Тестирование программы

Проведение тестирования на основе таблиц 1-7, показало, что программа выполнена верно, так как совпала с ожидаемым результатом.

## **2.4 Вывод по заданию**

Двунаправленный список, также известный как двусвязный список, представляет собой структуру данных, где каждый узел содержит данные и два указателя: на предыдущий и следующий узлы. Эта структура обеспечивает эффективное добавление и удаление элементов в любом месте списка, а также возможность перемещения как в прямом, так и в обратном направлении.

При использовании двунаправленного списка важно обновлять указатели на предыдущий и следующий узлы при добавлении и удалении элементов. Также необходимо учитывать краевые случаи, такие как вставка в начало или конец списка. При использовании итераторов следует учитывать направление движения.

Итак, двунаправленный список обеспечивает удобную структуру данных для операций вставки, удаления и перемещения как в прямом, так и в обратном направлении. Это особенно полезно при необходимости частого добавления и удаления элементов из середины списка или доступа к элементам в обратном порядке. Однако, при частом поиске элементов в списке другие структуры данных, такие как хэш-таблицы или деревья, могут быть более эффективными.

# 

# **3 ВЫВОДЫ**

Подводя итоги практической работы, можно сказать, что достигнуты следующие цели:

* Овладение управлением двунаправленным списком в программах на языке C++.
* Получение практических навыков работы с этой структурой данных.
* Анализ структуры и операций, связанных с индивидуальным вариантом.
* Реализация программы для работы со структурой данных в соответствии с индивидуальным вариантом.
* Разработка функций для операций с этой структурой данных.
* Создание меню для выбора способа ввода и реализации функций.
* Оценка сложности выполнения одной из дополнительных операций.
* Проведение тестирования программы с различными данными и способами ввода.

Таким образом, можно считать, что основная цель работы – овладение знаниями и навыками работы с двунаправленным списком в С++ – достигнута.

# **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).