МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение высшего образования

«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Отчет по выполнению практического задания 7

Тема:

Двунаправленные динамические списки

Выполнил студент Цемкало А. Р.

Фамилия И. О.

группа ИКБО-10-20

Москва 2021

СОДЕРЖАНИЕ

Цель. Получение знаний и практических навыков управления двунаправленным списком в программе.

Вариант 10.

1. Постановка задачи.

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

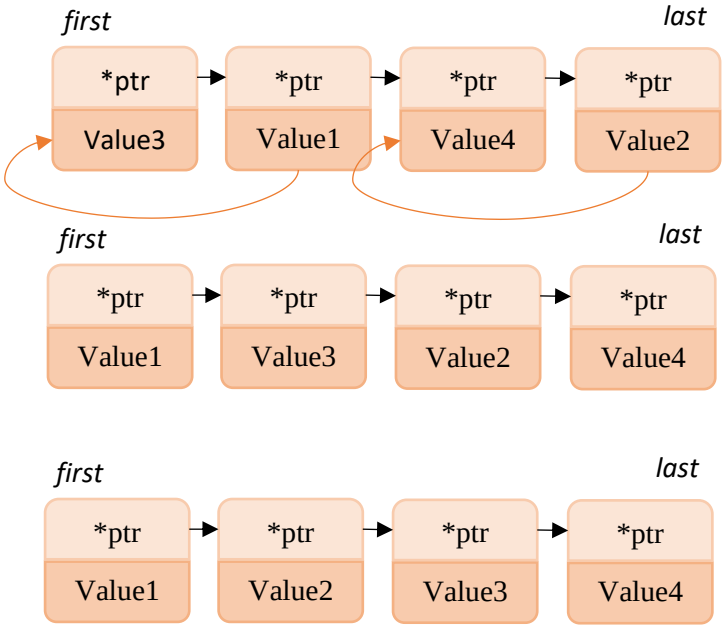
2. Определение списка операций над списком, которые выявлены в процессе исследования задач дополнительного задания.

2.1. Определить структуру узла двунаправленного списка в соответствии с вариантом.

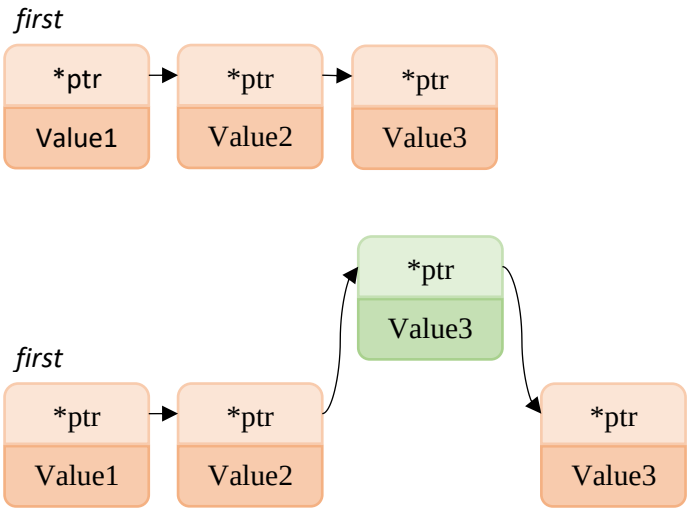
```
struct Node {
	string car_model;
	string country;
	int year;
	int price;
	string date_of_sale;
	Node* next;
	Node* previous;
	Node(string _car_model, string _country, int _year, int _price, string _date_of_sale
= "") : next(nullptr), previous(nullptr), car_model(_car_model), country(_country),
year(_year), price(_price), date_of_sale(_date_of_sale) {}
};
```

2.2. Изобразить (рисунок) для каждой операции полученного списка процесс выполнения операции на существующем однонаправленном списке.

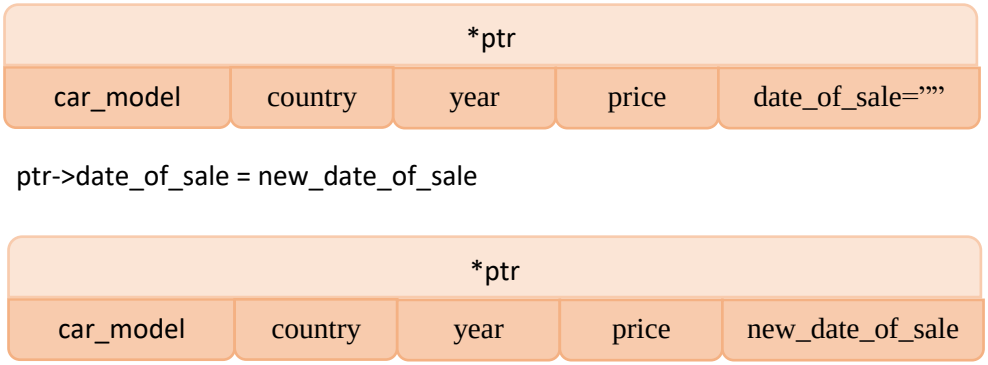
Упорядочить созданный список из n узлов так, чтобы узлы были упорядочены по стране изготовителю (будут сформированы подсписки по стране).



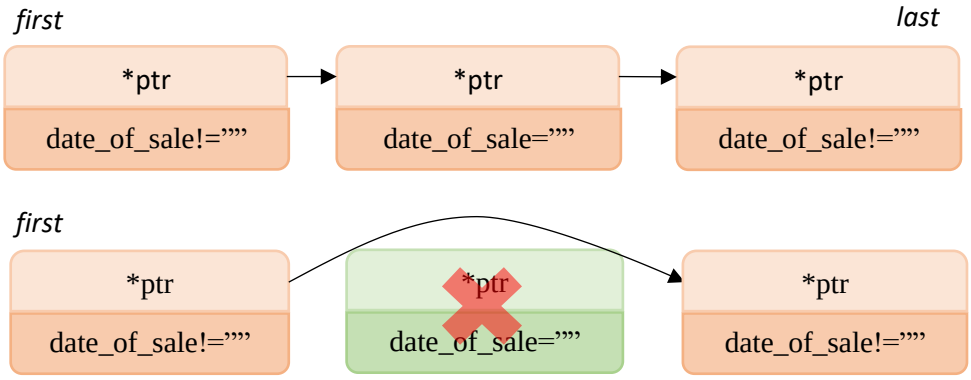
Вставить новый узел со сведениями об автомобиле какой-то страны в начало своего подсписка.



Установить дату продажи проданному автомобилю.



Удалить все узлы по проданным автомобилям.



2.3. Изобразите структуру данных, которая будет использоваться в операциях.

```
struct list {
	Node* first;
	Node* last;
	int list_size = 0;
	list() : first(nullptr), last(nullptr) {}
};
```

2.4. Привести алгоритм выполнения операции

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

Если список пустой, создаём первый узел. Прекращение работы функции.

Если в данной стране не было выпущено автомобилей, новый автомобиль добавляется в конец списка.

Иначе новый узел вставляется перед первым найденным узлом, у которого в данных о стране записана нужная нам: в указатель предыдущем узле на новый узел записывается указатель на новый, в указатель нового на предыдущей –

указатель текущего на предыдущей, в указатель нового на следующий – указатель на старый, в указатель текущего узла на предыдущий – указатель на новый узел.

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

Используется алгоритм ускоренной сортировки «Прямое слияние».

2.5. Привести таблицу тестов для тестирования каждой операции

Функция упорядочивания созданного списка из n узлов так, чтобы узлы были упорядочены по стране изготовителю (будут сформированы подсписки по стране).

№	Входные данные	Ожидаемый список	Список после выполнения программы
1	m1 china 1991 10 m2 germany 1983 20 m3 italy 2001 30 m4 germany 1998 40 m5 italy 1995 50	m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50	m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50
2	m1 New Zeland 1991 10 m2 Switzerland 1983 20 m3 Japan 2001 30 m4 Germany 1998 40 m5 Italy 1995 50	m4 Germany 1998 40 m5 Italy 1995 50 m3 Japan 2001 30 m1 New Zeland 1991 10 m2 Switzerland 1983 20	m4 Germany 1998 40 m5 Italy 1995 50 m3 Japan 2001 30 m1 New Zeland 1991 10 m2 Switzerland 1983 20

Функция вставки нового узла со сведениями об автомобиле какой-то страны в начало своего подсписка.

№	Входные данные	Ожидаемый список	Список после выполнения программы
1	Старый отсортированный список: m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50 Добавляемый узел: m7 germany 2021 70	m1 china 1991 10 m7 germany 2021 70 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50	m1 china 1991 10 m7 germany 2021 70 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50
2	Старый отсортированный список: m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50 Добавляемый узел: m7 austria 2021 70	m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50 m7 austria 2021 70	m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50 m7 austria 2021 70

Функция установки даты продажи проданному автомобилю.

№	Входные данные	Ожидаемый список	Список после выполнения программы
1	Старый отсортированный список: m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50	m1 china 1991 10 m2 germany 1983 20 today m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50	m1 china 1991 10 m2 germany 1983 20 today m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50
2	Старый отсортированный список: m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50	m1 china 1991 10 m2 germany 1983 20 today m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50 21.04.2021	m1 china 1991 10 m2 germany 1983 20 today m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50 21.04.2021

Функция удаления всех узлов по проданным автомобилям.

№	Входные данные	Ожидаемый список	Список после выполнения программы
1	m1 china 1991 10 20.04.2021 m2 germany 1983 20 today m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50 21.04.2021	m4 germany 1998 40 m3 italy 2001 30	m4 germany 1998 40 m3 italy 2001 30
2	m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50	m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50	m1 china 1991 10 m2 germany 1983 20 m4 germany 1998 40 m3 italy 2001 30 m5 italy 1995 50

3. Код программы

Основной файл task_7.cpp

```
#include <iostream>
#include "task_7.h"
using namespace std;

bool list_is_empty(list* L);
Node* find_first_coming_value(list* L, string type, string value);
Node* find_first_coming_value(list* L, string type, int value);
Node* add(list* L, string _car_model, string _country, int _year, int _price, string
_date_of_sale = " ");
Node* add_by_country(list* L, string _car_model, string _country, int _year, int _price,
string _date_of_sale = " ");
void print(list* L, string direction);
void set_date_of_sale(list* L, Node* p, string date);
void delete_sold_automobiles(list* L);
void sort_list_by_country(list* L, int n);

int main() {
	list L;
	cout << "The list is created" << endl;
	if (list_is_empty(&L)) {
		cout << "The list is empty" << endl;
	}
	else {
		cout << "The list is not empty" << endl;
	}
```

```
	cout << endl;

	add(&L, "model1", "china", 1991, 10);
	add(&L, "model2", "germany", 1983, 20);
	add(&L, "model3", "italy", 2001, 30);
	add(&L, "model4", "germany", 1998, 40);
	add(&L, "model5", "italy", 1995, 50);
	cout << "Some symbols are added to the list." << endl << "This is our list left to
right:" << endl;
	print(&L, "left_to_right");
	cout << endl;
	cout << "This is our list right to left:" << endl;
	print(&L, "right_to_left");

	cout << endl;
	if (list_is_empty(&L)) {
		cout << "The list is empty" << endl;
	}
	else {
		cout << "The list is not empty" << endl;
	}
	cout << endl;

	cout << "This is our list sorted by country: " << endl;
	sort_list_by_country(&L, L.list_size);
	print(&L, "left_to_right");
	cout << endl;

	cout << "New car is added to the list. This is our list left to right:" << endl;
	add_by_country(&L, "model7", "germany", 2021, 70);
	print(&L, "left_to_right");

	cout << endl;
	cout << "Some cars are sold: " << endl;
	set_date_of_sale(&L, find_first_coming_value(&L, "year", 1983), "today");
	set_date_of_sale(&L, find_first_coming_value(&L, "price", 50), "21.04.2021");
	print(&L, "left_to_right");

	cout << endl;
	cout << "Let's delete sold cars from our list:" << endl;
	delete_sold_automobiles(&L);
	print(&L, "left_to_right");

	return 0;
}
```

Заголовочный файл task_7.h

```
#ifndef __Task7_H
#define __Task7_H
#include <iostream>
using namespace std;

struct Node {
	string car_model;
	string country;
	int year;
	int price;
	string date_of_sale;
	Node* next;
	Node* previous;
	Node(string _car_model, string _country, int _year, int _price, string _date_of_sale
= "") : next(nullptr), previous(nullptr), car_model(_car_model), country(_country),
year(_year), price(_price), date_of_sale(_date_of_sale) {}
};

struct list {
	Node* first;
	Node* last;
	int list_size = 0;
	list() : first(nullptr), last(nullptr) {}
};

#endif
```

Отдельный файл с функциями task_7_extra_functions.cpp

```cpp
#include <iostream>
#include "task_7.h"
using namespace std;

bool list_is_empty(list* L) {
	return L->first == nullptr && L->last == nullptr;
}

Node* find_first_coming_value(list* L, string type, string value) {
	Node* p = L->first;
	bool f = true;
	while (f) {
		if (type == "car_model" && p->car_model == value) {
			return p;
		}
		else if (type == "country" && p->country == value) {
			return p;
		}
		else if (type == "date_of_sale" && p->date_of_sale == value) {
			return p;
		}
		f = p != L->last;
		p = p->next;
	}
	return NULL;
}

Node* find_first_coming_value(list* L, string type, int value) {
	Node* p = L->first;
	bool f = true;
	while (f) {
		if (type == "year" && p->year == value) {
			return p;
		}
		else if (type == "price" && p->price == value) {
			return p;
		}
		f = p != L->last;
		p = p->next;
	}
}

Node* add(list* L, string _car_model, string _country, int _year, int _price, string
_date_of_sale) {
	L->list_size++;
	Node* p = new Node(_car_model, _country, _year, _price, _date_of_sale);
	if (list_is_empty(L)) {
		L->first = p;
		L->last = p;
		return p;
	}

	L->last->next = p;
	p->previous = L->last;
	L->last = p;
	return p;
}

Node* add_by_country(list* L, string _car_model, string _country, int _year, int _price,
string _date_of_sale) {
	L->list_size++;
	Node* p = new Node(_car_model, _country, _year, _price, _date_of_sale);
	if (list_is_empty(L)) {
```

```
			L->first = p;
			L->last = p;
			return p;
		}
		Node* old_p = find_first_coming_value(L, "country", _country);
		if (old_p == NULL) {
			L->last->next = p;
			p->previous = L->last;
			L->last = p;
			return p;
		}
		old_p->previous->next = p;
		p->previous = old_p->previous;
		p->next = old_p;
		old_p->previous = p;
		return p;
}

void print(list* L, string direction) {
		if (list_is_empty(L)) return;
		if (direction == "left_to_right") {
			Node* p = L->first;
			while (p) {
				cout << p->car_model << " " << p->country << " " << p->year << " " <<
p->price;
				if (p->date_of_sale != "") {
					cout << " " << p->date_of_sale;
				}
				cout << endl;
				p = p->next;
			}
		}
		else if (direction == "right_to_left") {
			Node* p = L->last;
			while (p) {
				cout << p->car_model << " " << p->country << " " << p->year << " " <<
p->price;
				if (p->date_of_sale != "") {
					cout << " " << p->date_of_sale;
				}
				cout << endl;
				p = p->previous;
			}
		}
}

void set_date_of_sale(list* L, Node* p, string date) {
		p->date_of_sale = date;
}

void delete_sold_automobiles(list* L) {
		Node* p = L->first;
		bool f = true;
		while (f) {
			if (p->date_of_sale != " ") {
				if (p->previous != NULL) {
					p->previous->next = p->next;
				}
				else {
					L->first = p->next;
				}
				if (p->next != NULL) {
					p->next->previous = p->previous;
				}
				else {
```

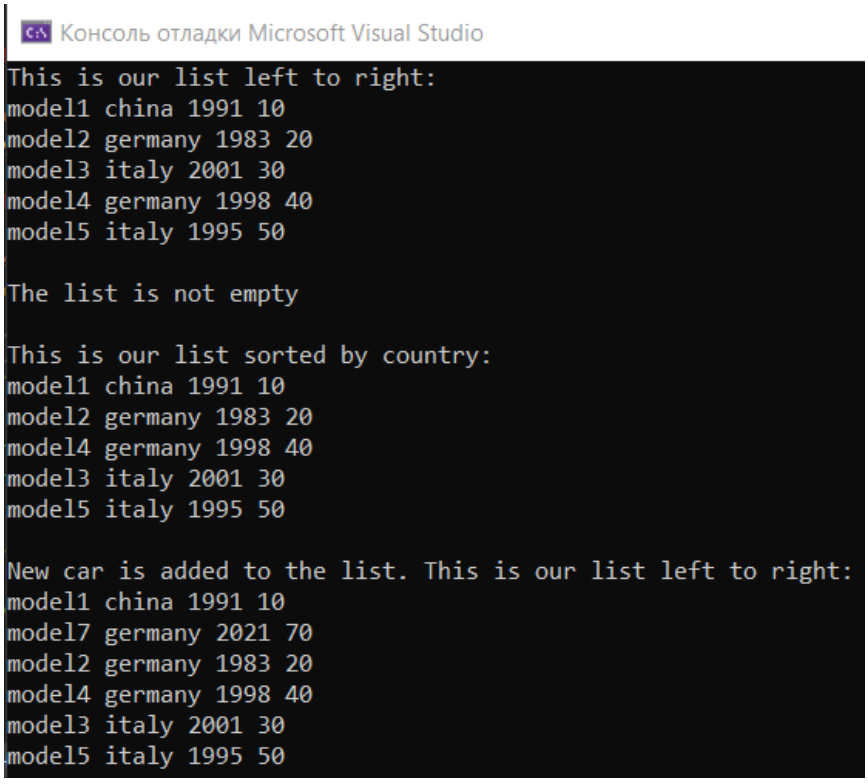
```
                        L->last = p->previous;
                        return;
                    }
                }
                f = p != L->last;
                p = p->next;
        }
}

void sort_list_by_country(list* L, int n) {
        if (n == 0 || n == 1) {
                return;
        }
        int left_len = n / 2;
        int right_len = n - left_len;
        list left;
        list right;
        Node* p = L->first;
        for (int i = 0; i < left_len; i++) {
                add(&left, p->car_model, p->country, p->year, p->price, p->date_of_sale);
                p = p->next;
        }
        for (int i = 0; i < right_len; i++) {
                add(&right, p->car_model, p->country, p->year, p->price, p->date_of_sale);
                p = p->next;
        }
        sort_list_by_country(&left, n / 2);
        sort_list_by_country(&right, n - n / 2);
        int x = 0, y = 0, k = 0;
        Node* p_x = left.first, * p_y = right.first;
        list new_L;
        while (x < left_len && y < right_len) {
                if (p_x->country <= p_y->country) {
                        add(&new_L, p_x->car_model, p_x->country, p_x->year, p_x->price, p_x-
>date_of_sale);
                        p_x = p_x->next;
                        x++;
                }
                else {
                        add(&new_L, p_y->car_model, p_y->country, p_y->year, p_y->price, p_y-
>date_of_sale);
                        p_y = p_y->next;
                        y++;
                }
        }
        while (x < left_len) {
                add(&new_L, p_x->car_model, p_x->country, p_x->year, p_x->price, p_x-
>date_of_sale);
                p_x = p_x->next;
                x++;
        }
        while (y < right_len) {
                add(&new_L, p_y->car_model, p_y->country, p_y->year, p_y->price, p_y-
>date_of_sale);
                p_y = p_y->next;
                y++;
        }
        (*L) = new_L;
}
```

4. Результат тестирования программы: скриншоты выполнения каждой операции.

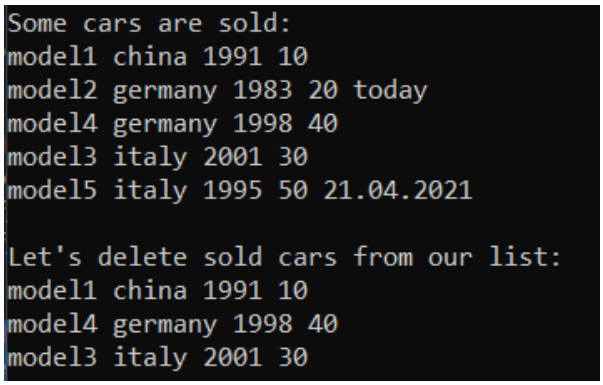
Функция упорядочивания созданного списка из n узлов так, чтобы узлы были упорядочены по стране изготовителю (будут сформированы подсписки по стране)

Функция вставки нового узла со сведениями об автомобиле какой-то страны в начало своего подсписка

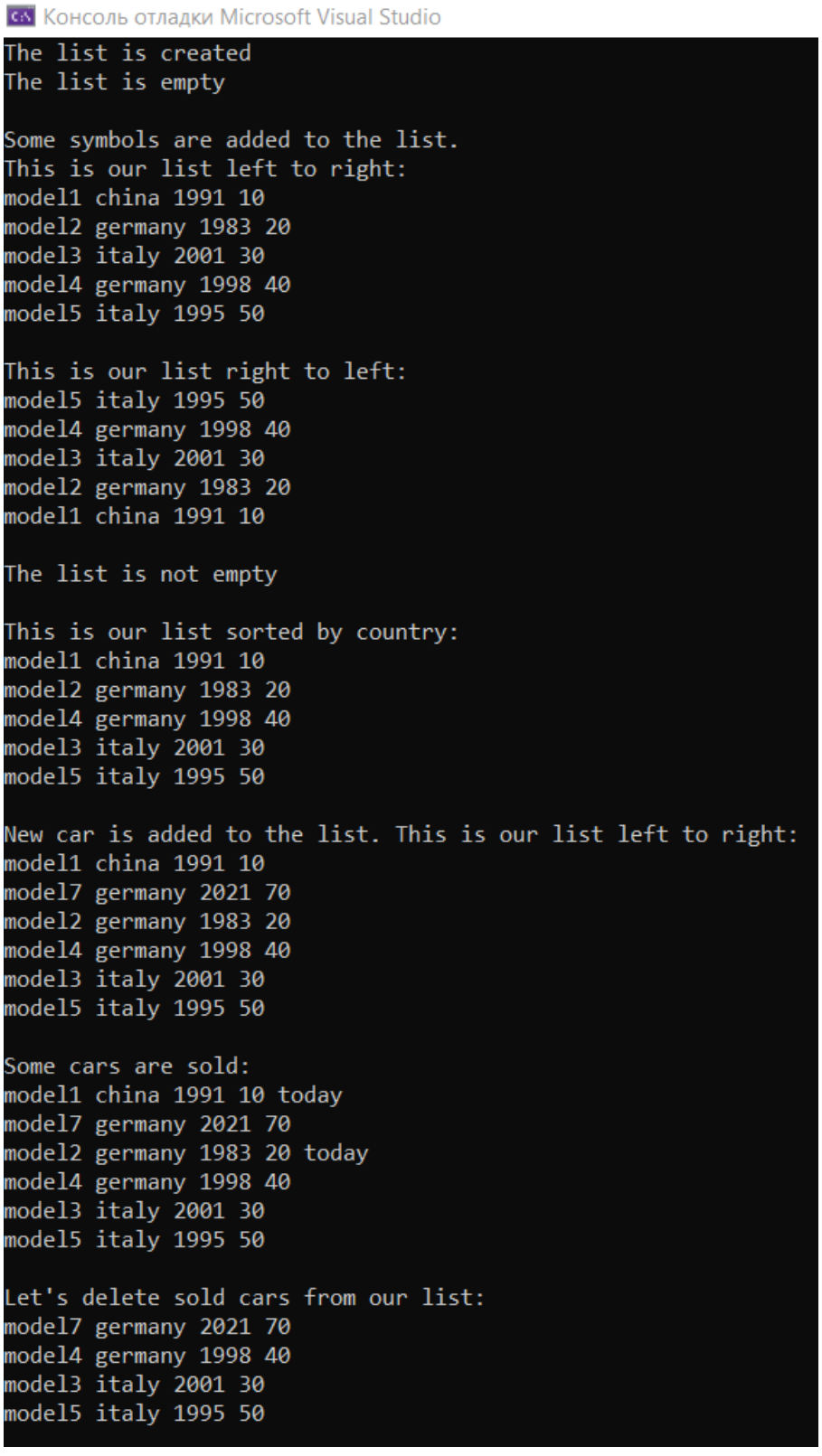


Функция установки даты продажи проданному автомобилю

Функция удаления всех узлов по проданным автомобилям



Работа всех функций



ВЫВОДЫ

В ходе выполнения задания получены знания и практические навыки управления двунаправленным списком. Разработаны:

1. Структура узла списка
2. Функция для создания исходного списка, используя функцию вставки нового узла после первого узла;
3. Функция вывода списка в двух направлениях;
4. Функция поиска узла с заданным значением;
5. Функция упорядочивания созданного списка из n узлов так, чтобы узлы были упорядочены по стране изготовителю;
6. Функция вставки нового узла со сведениями об автомобиле какой-то страны в начало своего подсписка.
7. Функция установки даты продажи проданному автомобилю.
8. Функция удаления всех узлов по проданным автомобилям.

Тестирования всех операций пройдены успешно.

СПИСОК ИНФОРМАЦИОННЫХ ИСТОЧНИКОВ

1. Процедурное программирование Языки программирования – Сайт lizochekk! [Электронный ресурс]: URL: https://lizochekk.jimdofree.com/программирование/
2. Документация по Microsoft C/C++ | Microsoft Docs – [Электронный ресурс] URL: https://docs.microsoft.com/ru-ru/cpp/?view=msvc-160
3. Все публикации подряд / Хабр – [Электронный ресурс] URL: https://habr.com/ru/