Выполнил

студент КТбо1-4 \_\_\_\_\_\_\_\_\_\_\_\_\_ А. А. Воронов

Принял

ст. преподаватель каф. САПР \_\_\_\_\_\_\_\_\_\_\_\_\_ В. И. Данильченко

Таганрог 2024

ЛАБОРАТОРНАЯ РАБОТА №8

по дисциплине «Алгоритмизация и программирование»

**«Реализация связанного списка»**

Министерство науки и высшего образования Российской Федерации

Федеральное государственное автономное образовательное учреждение высшего образования

«ЮЖНЫЙ ФЕДЕРАЛЬНЫЙ УНИВЕРСИТЕТ»

(ФГАОУ ВО «ЮФУ»)

1. **Цель лабораторной работы:** Изучить работу связанного списка, реализовать основные методы работы с ним.
2. **Задача:** Написать программу, в которой реализован односвязный список. Он должен иметь следующие методы:

1. Включить новый элемент в конец списка.

2. Включить новый элемент на заданное пользователем место списка.

3. Включить новый элемент после элемента с заданной информационной частью.

4. Включить новый элемент перед элементом с заданной информационной частью.

5. Включить новый элемент в середину списка.

6. Исключить элемент из середины списка.

7. Исключить элемент с заданной информационной частью.

8. Исключить элемент из конца списка.

9. Исключить элемент из заданного пользователем места списка.

10. Исключить элемент из головы списка.

11. Выводить данные списка.

Список должен использоваться для организации учета автомобилей в ГАИ.

1. **Ход работы:** я создал проект-консольное приложение в Visual Studio 2022. Реализовал односвязный список. Данные об автомобилях передаются в структуре car\_data. Для обработки запросов пользователя из консоли используется конструкция switch-case. Односвязный список имеет следующие методы: push\_back — добавление в конец списка нового узла, push\_front — добавление нового узла в начало списка, insert — добавление нового узла в список по определенному индексу, eliminate — удаление узла по индексу, pop\_back — удаление последнего узла списка, pop\_front — удаление первого узла списка, get\_by\_idx — получение узла по определенному индексу, print вывод информации из списка, scan — поиск индекса узла по номеру автомобиля, scan\_middle —поиск индекса центра, creator — создание элемента структуры car\_data. Для ввода данных об автомобиле использовалась функция assign.

**Программный код:**

#include <iostream>

#include <string>

#include <fstream>

using namespace std;

struct car\_data {

string owner\_surname, owner\_name, owner\_patronymic, car\_model, car\_number;

void p\_print() {

cout

<< "\n"

<< "Owner: \n"

<< "Surname: " << owner\_surname << "\n"

<< "Name: " << owner\_name << "\n"

<< "Patronymic: " << owner\_patronymic << "\n\n"

<< "Car: " << "\n"

<< "Model: " << car\_model << "\n"

<< "Number: " << car\_number << "\n"

<< "================================== \n\n";

}

};

struct Node {

car\_data val;

Node\* next;

Node(car\_data val) {

this->val = val;

this->next = NULL;

}

};

car\_data assign(car\_data x);

struct list { //односвязный список

Node\* first, \* last;

list() : first(nullptr), last(nullptr) {}

// General list methods

void push\_back(car\_data data) { // 1 добавляет в конец списка

Node\* node = new Node(data);

if (first == NULL) first = node;

if (last != NULL) last->next = node;

last = node;

}

void push\_front(car\_data data) { //добавляет в начало списка

Node\* node = new Node(data);

node->next = first;

first = node;

if (last = NULL) last = node;

}

void insert(car\_data data, int idx) { // 2, 3, 4, 5 добавляет по индексу

Node\* left = get\_by\_idx(idx-1);

if (left == NULL) return push\_front(data);;

Node\* right = left->next;

Node\* node = new Node(data);

left->next = node;

node->next = right;

if (right == NULL) last = node;

}

void eliminate(int idx) { // 6, 7, 9 удаление узла по индексу

if (idx < 0) return;

if (idx == 0) {

pop\_front();

return;

}

Node\* left = get\_by\_idx(idx - 1);

Node\* node = left->next;

if (node == NULL) return;

Node\* right = node->next;

left->next = right;

if (node == last) last = left;

delete node;

}

void pop\_back() { // 8 удаляет последний объект

if (last == NULL) return;

if (first == last) {

delete last;

first = last = NULL;

return;

}

Node\* node = first;

for (; node->next != last; node = node->next);

node->next = NULL;

delete last;

last = node;

}

void pop\_front() { // 10 удаляет первый объект

if (first == NULL) return;

if (first == last) {

delete last;

first = last = NULL;

return;

}

Node\* node = first;

first = node->next;

delete node;

}

// Additional list methods

Node \* get\_by\_idx(int idx) {

if (idx < 0) return NULL;

Node\* node = first;

int n = 0;

while (node && n != idx && node->next) {

node = node->next;

n++;

}

if (n == idx) return node;

else return NULL;

}

void print() {

Node\* node = first;

for (; node != NULL; node = node->next) {

node->val.p\_print();

}

}

int scan(string cnum) { //поиск номеру автомобиля

Node\* node = first;

int idx = 0;

while(node!=NULL){

if (node->val.car\_number == cnum) return idx;

idx++;

node = node->next;

}

return -1;

}

int scan\_middle() {

Node\* node = first;

if (first == NULL) return 0;

int length = 0;

while (node != NULL) {

length++;

node = node->next;

}

return length / 2;

}

car\_data creator() {

car\_data p;

p = assign(p);

return p;

}

};

car\_data assign(car\_data x) {

getline(cin, x.owner\_surname, ' ');

x.owner\_surname = x.owner\_surname.substr(1, x.owner\_surname.size());

getline(cin, x.owner\_name, ' ');

getline(cin, x.owner\_patronymic, ' ');

getline(cin, x.car\_model, ' ');

getline(cin, x.car\_number);

return x;

}

//----------------------------------------------------

int main() {

list lst;

int n = -1;

while (n != 0) {

cout

<< "\nChoose action:\n"

<< "0. Exit\n"

<< "1. Add to the end\n"

<< "2. Add by index\n"

<< "3. Add after another depending on data (car number)\n"

<< "4. Add before another depending on data (car number)\n"

<< "5. Add to the middle\n"

<< "6. Delete from the middle\n"

<< "7. Delete depending on data\n"

<< "8. Delete last\n"

<< "9. Delete by index\n"

<< "10. Delete first \n"

<< "11. Print \n\n"

<< "Your choise is: ";

cin >> n;

switch (n) {

case 0:

break;

case 1: {

cout << "\nEnter data: Surname/Name/Patronymic/Model/Number: ";

lst.push\_back(lst.creator());

break;

}

case 2: {

cout << "\nEnter data: Surname/Name/Patronymic/Model/Number: ";

car\_data car = lst.creator();

string stridx;

int idx;

cout << "\nEnter Index: ";

cin >> stridx;

idx = stoi(stridx);

lst.insert(car, idx);

break;

}

case 3: {

cout << "\nEnter data: Surname/Name/Patronymic/Model/Number: ";

car\_data car = lst.creator();

string cnum;

cout << "\nEnter Number: ";

cin >> cnum;

int idx = lst.scan(cnum);

lst.insert(car, idx + 1);

break;

}

case 4: {

cout << "\nEnter data: Surname/Name/Patronymic/Model/Number: ";

car\_data car = lst.creator();

string cnum;

cout << "\nEnter Number: ";

cin >> cnum;

int idx = lst.scan(cnum);

lst.insert(car, idx);

break;

}

case 5: {

cout << "\nEnter data: Surname/Name/Patronymic/Model/Number: ";

car\_data car = lst.creator();

lst.insert(car, lst.scan\_middle());

break;

}

case 6:

if (lst.scan\_middle() == 0) lst.pop\_back();

else if(lst.scan\_middle() % 2 == 0) lst.eliminate(lst.scan\_middle() - 1);

else lst.eliminate(lst.scan\_middle());

break;

case 7: {

cout << "\nEnter Number: ";

string cnum;

cin >> cnum;

int idx = lst.scan(cnum);

if (idx == -1) {

cout << "\n Not Found\n";

break;

}

lst.eliminate(idx);

break;

}

case 8: {

lst.pop\_back();

break;

}

case 9: {

cout << "\nEnter Index: ";

string stridx;

cin >> stridx;

int idx = stoi(stridx);

lst.eliminate(idx);

break;

}

case 10: {

lst.pop\_front();

break;

}

case 11:

cout << "\n";

lst.print();

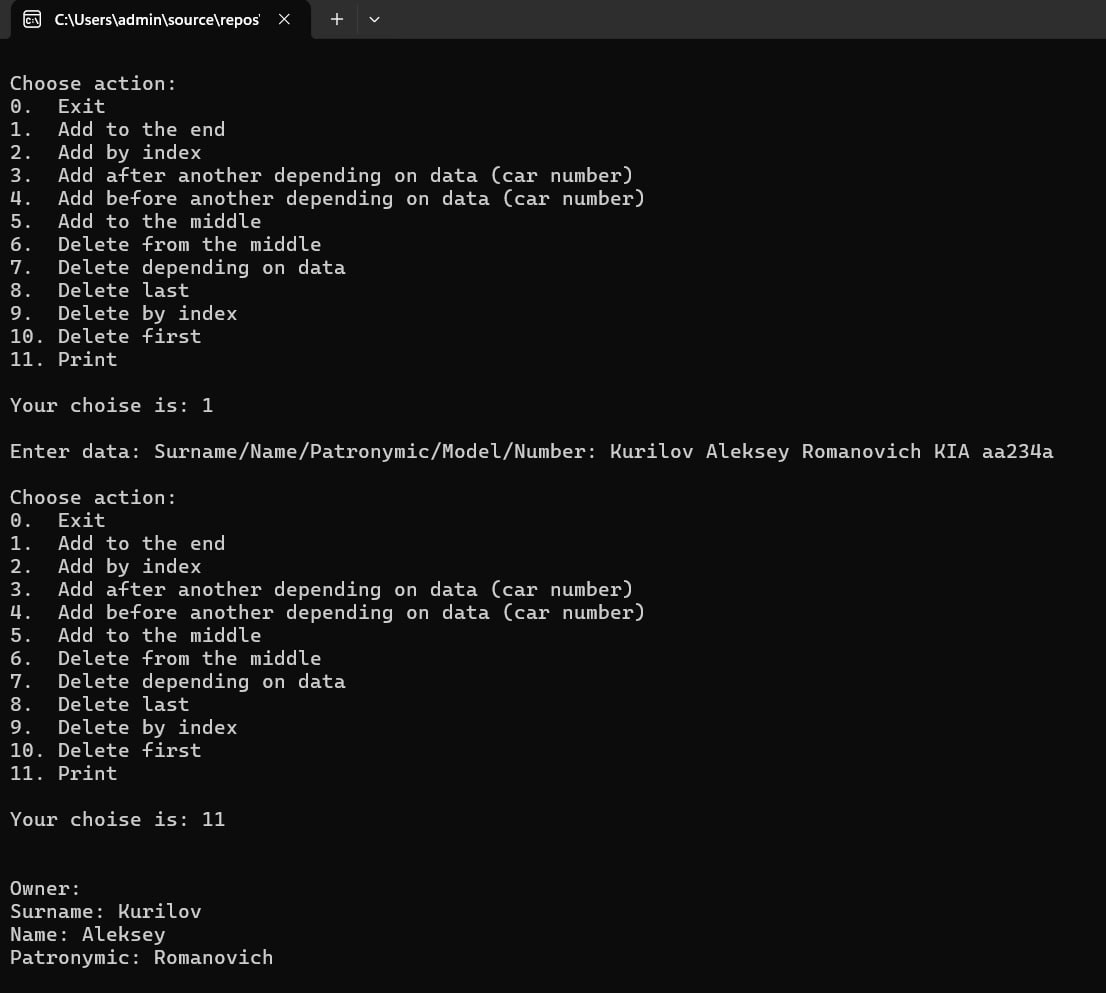
break;

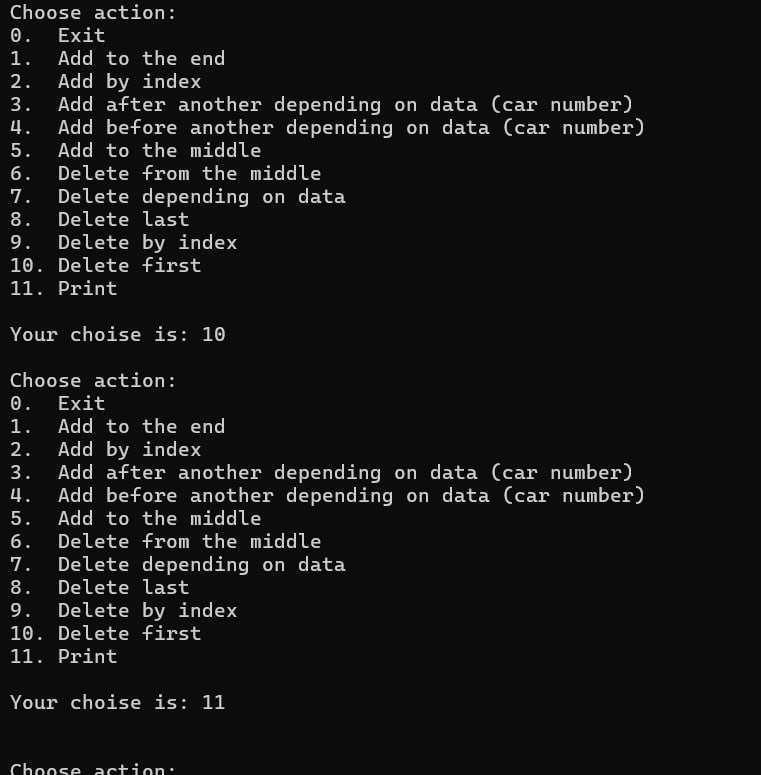
}

}

}

**Пример работы программы:**





**4. Вывод:** в ходе работы были выполнены все цели и задачи, было выяснено, как работать со списком структур и создавать односвязный список в языке С++, было реализовано: а) дополнение существующего списка структур новыми структурами; б) поиск структуры с заданным значением выбранного элемента; в) вывод на экран содержимого списка структур; г) упорядочение массива структур по заданному полю (элементу).

**5. Источники:**

<https://habr.com/ru/sandbox/153128/>

<https://itnotesblog.ru/note/odnosvyaznyj-spisok-na-cpp>

https://learn.microsoft.com/ru-ru/cpp/cpp/switch-statement-cpp?view=msvc-170