МИНОБРНАУКИ РОССИИ

Федеральное государственное бюджетное образовательное учреждение
высшего образования
«МИРЭА – Российский технологический университет»

РТУ МИРЭА

Институт Информационных технологий

Кафедра Математического обеспечения и стандартизации информационных технологий

ОТЧЕТ ПО ПРАКТИЧЕСКОЙ РАБОТЕ № 10

по дисциплине

«Структуры и алгоритмы обработки данных»

Тема: «Поиск в тесте. Алгоритмы»

Выполнил студент группы ИКБО-18-22 Ракитин В.А.

Принял преподаватель Филатов А.С.

Лабораторная работа выполнена «__»_____202__ г. *(подпись студента)*

«Зачтено» «__»_____202__ г. *(подпись руководителя)*

Москва 2023

1. Цель работы

Получить знания и навыки применения алгоритмов поиска в тексте подстрок.

2. Постановка задачи

Задание 1. Разработать и реализовать алгоритм поиска в тексте

1. Включить в этап «Решение» описание алгоритма рассматриваемого метода. Разобрать алгоритм на примере. Подсчитать количество сравнений для успешного поиска первого вхождения образца в текст и безуспешного поиска.
2. Разработать и отладить функции для реализации алгоритма.
3. Сформировать таблицу тестов с указанием успешного и неуспешного поиска, используя большой по объему текст, и образец различного объема. Включить ее в этап тестирования
4. Разработать и реализовать программу тестирования алгоритма.
5. Оценить практическую сложность алгоритма в зависимости от длины текста и длины образца и отобразить результаты в таблице.
6. Оформить отчет, включив в него этапы разработки каждой задачи варианта. Сравнить эффективность алгоритма, как практическую, так и теоретическую.

Задание 2. Разработать алгоритм и функцию поиска образца в тексте с применением регулярных выражений для второй задачи варианта

1. Разработать регулярное выражение в соответствии с задачей варианта.
2. Разработать функцию, реализующую проверку входной строки на соответствие регулярному выражению или ее модификацию в соответствии с индивидуальным вариантом.
3. Выполнить тестирование на разработанных тестах.
4. Оформить отчет, представив результаты по пунктам задания.

Вариант №21. Условие задания:

Упражнение 1	Дан пакет из n документов. Каждый документ – это текст протокола регистрации ДТП. В протоколе указан номер автомобиля, участвующего в ДТП. Российские автомобильные номера, формируемые по формату: «буква - три цифры - две буквы - код региона». Причем код региона может быть двух или трехзначным, а в качестве букв применяются только те, что похожи внешне на латиницу. Определить сколько нарушений у владельца автомобиля с заданным номером. Например, дан текст: БМВ Х3 В123АУ777 черн. Е83 2.0d, результат В123АУ777
Упражнение 2	Определить, является ли строка корректной датой с 1000 года. Учесть количество дней в месяцах. Считать, что в феврале всегда 29 дней

3. Решение

Регулярные выражения (Regular expressions) — это формальный язык описания паттернов (образцов текста) для поиска и замены текста, основанный на использовании подстановочных и метасимволов. Регулярные выражения являются стандартом сопоставления с шаблоном для синтаксического анализа и изменения строк, и позволяют пользователю выразить, как компьютерная программа должна искать указанный шаблон в тексте, а затем, что она должна делать, когда найдено каждое совпадение с данным шаблоном. Для пользования регулярными выражениями, необходимо подключить библиотеку <regex>.

Для решения первого упражнения была написана функция generateFile, которая генерирует данные об автомобиле. На вход функция получает целочисленное значение – количество автомобилей, которые участвовали в ДТП. Далее программа создаёт данные об автомобилях и записывает эти данные в файл «protocol.txt».

```
void generateFile(int numberCar) {
	ofstream fout(fileName, ios::app);
	if (fout.is_open()) {
		char Auto[][8] = { "CITROEN", "fERRARI", "GENESIS", "PEUGEOT",
"RENAULT", "CHANGAN", "BUGATTI", "BENTLEY",
		"MAYBACH", "MCLAREN" };
		char number[][10] = {"A324TO799", "P308CC199", "X023TO109",
"H991KM199", "M555MM999", "T811BM238", "A753CB729",
		"H608TE899", "H888TT299", "O108TM877", "B101CC450", "M199MM199",
"P028AB308", "T909PP109", "T880AP709",
		"T111PT098", "A601HH023", "B221CC777", "A239BP078", "T237TP889",
"T303PE308", "O322PT230", "E443TT504" };
		for (int i = 0; i < numberCar; i++) {
			int index1 = rand() % 10;
			int index2 = rand() % 23;

			fout << "Автомобиль " << Auto[index1] << " с номером " <<
number[index2] << " попал в ДТП" << endl;
		}
		int size = 50 * numberCar;
		fout.close();
		cout << "Данные успешно записаны" << endl << "Количество
символов - " << size << endl << endl;
	}
	else {
		cout << "Не удалось открыть файл" << endl << endl;
	}
}
```

Также для решения первого упражнения была написана функция coutFile, которая выводит содержимое файла.

```
void coutFile() {
	cout << "Список машин, учавствовавших в ДТП:" << endl;
	ifstream fin(fileName);
	if (fin.is_open()) {
		string line;
		while (getline(fin, line)) {
			cout << line << endl;
		}
		fin.close();
		cout << endl;
	}
	else {
		cout << "Не удалось открыть файл" << endl << endl;
	}
}
```

Также для решения первого упражнения была написана функция coutNumberCar, которая выводит количество автомобилей, участвовавших в ДТП. На вход функция получает два значения – номер автомобиля и количество операций сравнения. Далее программа выводит, сколько раз автомобиль с таким номером был в ДТП, время работы программы и количество операций сравнения.

```
void coutNumberCar(string inputNumber, long long int& C) {
	int sizeInputNumber = inputNumber.size();
	vector <int> prefix(sizeInputNumber, 0);
	prefix_function(inputNumber, prefix, C);

	int violations = 0; // количество нарушений у владельца автомобиля

	ifstream fin(fileName);
	if (fin.is_open()) {
		string line;
		auto start = std::chrono::steady_clock::now();
		while (getline(fin, line)) {
			// Ищем номер автомобиля в текущей строке
			int n = line.length();
			int j = 0;
			for (int i = 0; i < n; i++) {
				while (j > 0 && line[i] != inputNumber[j]) {
					C++;
					j = prefix[j - 1];
				}
				C++;
				if (line[i] == inputNumber[j]) {
					j++;
				}
				C++;
				if (j == inputNumber.length()) {
					violations++;
					j = prefix[j - 1];
				}
			}
		}
		auto end = std::chrono::steady_clock::now();
		std::chrono::duration<double> elapsed_seconds = end - start;
		cout << "Количество нарушений у автомобиля с номером " <<
inputNumber << " - " << violations << endl
			<< "Время работы программы - " << elapsed_seconds.count()
<< endl
			<< "C = " << C << endl << endl;
	}
	else {
		cout << "Не удалось открыть файл" << endl << endl;
	}
}
```

Также для решения первого упражнения была написана функция prefix_function, которая реализует метод поиска Кнута-Мориса-Пратта. На вход функция получает три значения: номер автомобиля, префиксный вектор и количество операций сравнения. Далее программа заполняет вектор «prefix» значениями префикс-функции для каждой позиции строки.

```
void prefix_function(string inputNumber, vector<int>prefix, long long int&
C) {
	int n = inputNumber.length();
	prefix[0] = 0;
	for (int i = 1; i < n; i++) {
		int j = prefix[i - 1];
		while (j > 0 && inputNumber[i] != inputNumber[j]) {
			C++;
```

```
                j = prefix[j - 1];
            }
            C++;
            if (inputNumber[i] == inputNumber[j]) {
                j++;
            }
            prefix[i] = j;
        }
}
```

Для решения второго упражнения была написана функция isValidDate, которая проверяет, является ли строка корректной датой в формате "dd.mm.yyyy". На вход функция получает нашу дату, которую надо проверить. Далее программа использует регулярное выражение для проверки соответствия формату. Если строка соответствует этому формату, функция возвращает true, в противном случае – false.

```
bool isValidDate(string str) {
    regex date("^((0[1-9]|[12]\\d|3[01])\\.(0[1-9]|1[0-2])\\.([1-
9]\\d{3}))$");
    // шаблон даты: dd.mm.yyyy

    return regex_match(str, date);
}
```

При запуске программы первого упражнения, пользователь видит пользовательское меню, где надо выбрать, какую задачу надо решить.

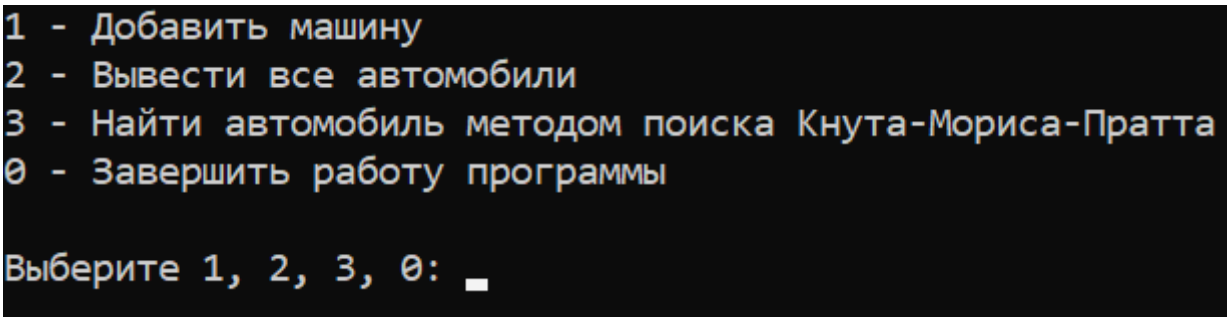

Рисунок 1. Интерфейс программы первого упражнения

При запуске программы второго упражнения пользователь видит пользовательское меню, где ему надо ввести дату.

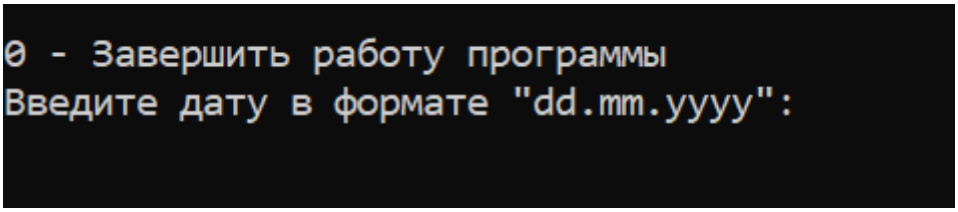

Рисунок 2. Интерфейс программы второго упражнения

4. Тестирование

Протестируем программой выполнение первого упражнения. Составим таблицу тестов с результатами успешного и неуспешного поиска.

Таблица 1. Данные об успешном поиске

n	**T(n)**	**T_a=f(N+M)**	**$T_э$=C**
100000	1.0178	5000009	10049355
200000	2.06938	10000009	20203579
300000	3.08526	15000009	30147570
400000	4.11703	20000009	40500461
500000	5.13402	25000009	50165083

Таблица 2. Данные о неуспешном поиске

n	**T(n)**	**T_a=f(N+M)**	**$T_э$=C**
100000	1.02581	5000009	10053690
200000	2.06557	10000009	20212371
300000	3.09531	15000009	30160564
400000	4.13967	20000009	40535503
500000	5.15434	25000009	50267771

```
1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 1
Сколько машин добавить?
100000
Данные успешно записаны
Количество символов - 5000000

1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 3
Введите номер автомобиля: P308CC199
Количество нарушений у автомобиля с номером P308CC199 - 4334
Время работы программы - 1.0178
C = 10049355

1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 3
Введите номер автомобиля: P221PT779
Количество нарушений у автомобиля с номером P221PT779 - 0
Время работы программы - 1.02581
C = 10053690
```

Рисунок 3. Решение программой первого упражнения

```
1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 1
Сколько машин добавить?
200000
Данные успешно записаны
Количество символов - 10000000

1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 3
Введите номер автомобиля: A239BP078
Количество нарушений у автомобиля с номером A239BP078 - 8791
Время работы программы - 2.06938
C = 20203579

1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 3
Введите номер автомобиля: A118AT899
Количество нарушений у автомобиля с номером A118AT899 - 0
Время работы программы - 2.06557
C = 20212371
```

Рисунок 4. Решение программой первого упражнения

```
1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 1
Сколько машин добавить?
300000
Данные успешно записаны
Количество символов - 15000000

1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 3
Введите номер автомобиля: P028AB308
Количество нарушений у автомобиля с номером P028AB308 - 12993
Время работы программы - 3.08526
C = 30147570

1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 3
Введите номер автомобиля: P330TP199
Количество нарушений у автомобиля с номером P330TP199 - 0
Время работы программы - 3.09531
C = 30160564
```

Рисунок 5. Решение программой первого упражнения

```
1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 1
Сколько машин добавить?
400000
Данные успешно записаны
Количество символов - 20000000

1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 3
Введите номер автомобиля: T237TP889
Количество нарушений у автомобиля с номером T237TP889 - 17521
Время работы программы - 4.11703
C = 40500461

1 - Добавить машину
2 - Вывести все автомобили
3 - Найти автомобиль методом поиска Кнута-Мориса-Пратта
0 - Завершить работу программы

Выберите 1, 2, 3, 0: 3
Введите номер автомобиля: T999TT999
Количество нарушений у автомобиля с номером T999TT999 - 0
Время работы программы - 4.13967
C = 40535503
```

Рисунок 6. Решение программой первого упражнения

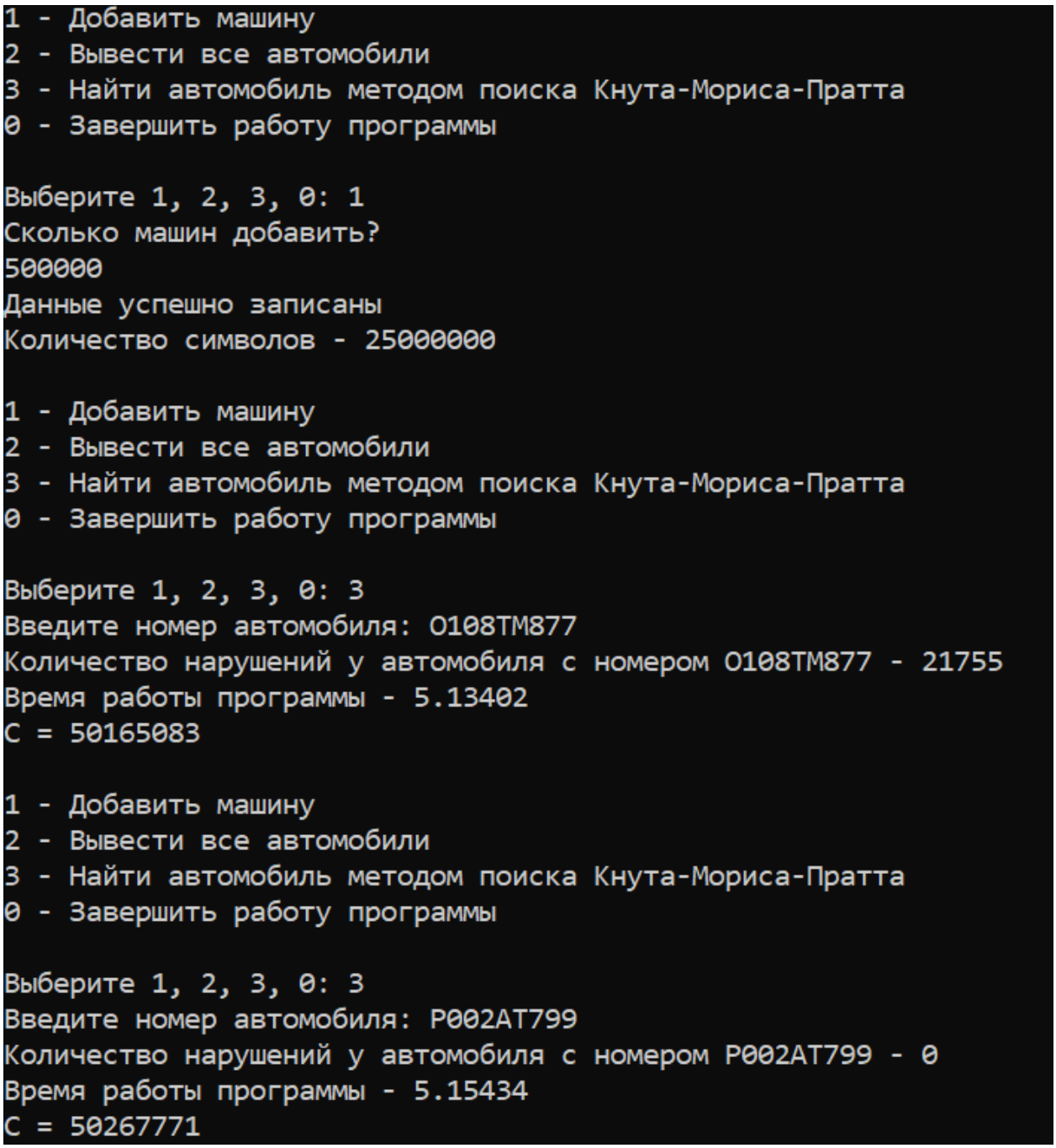

Рисунок 7. Решение программой первого упражнения

Протестируем программой выполнения второго упражнения. Будем вводить корректные и некорректные даты. На рисунке 8 видно, что программа выводит верные результаты.

```
0 - Завершить работу программы
Введите дату в формате "dd.mm.yyyy": 31.12.2004
Введенная дата является корректной датой с 1000 года.

0 - Завершить работу программы
Введите дату в формате "dd.mm.yyyy": 12.2.2030
Введенная строка не является корректной датой.

0 - Завершить работу программы
Введите дату в формате "dd.mm.yyyy": 32
Введенная строка не является корректной датой.

0 - Завершить работу программы
Введите дату в формате "dd.mm.yyyy": 32.02.1999
Введенная строка не является корректной датой.

0 - Завершить работу программы
Введите дату в формате "dd.mm.yyyy": 13.05.999
Введенная строка не является корректной датой.

0 - Завершить работу программы
Введите дату в формате "dd.mm.yyyy": 26.07.10000
Введенная строка не является корректной датой.
```

Рисунок 8. Решение программой второго упражнения

5. Вывод

В результате работы я:

1. Получил знания и навыки применения алгоритмов поиска в тексте
2. Получил знания и навыки применения регулярных выражений на языке программирования C++

6. Исходный код программы

```
#include <iostream>
#include <string>
#include <fstream>
#include <vector>
```

```cpp
#include <algorithm>
#include <chrono>
using namespace std;
const char* fileName = "protocol.txt";

void prefix_function(string inputNumber, vector<int>prefix, long long int&
C) {
	int n = inputNumber.length();
	prefix[0] = 0;
	for (int i = 1; i < n; i++) {
		int j = prefix[i - 1];
		while (j > 0 && inputNumber[i] != inputNumber[j]) {
			C++;
			j = prefix[j - 1];
		}
		C++;
		if (inputNumber[i] == inputNumber[j]) {
			j++;
		}
		prefix[i] = j;
	}
}

void coutNumberCar(string inputNumber, long long int& C) {
	int sizeInputNumber = inputNumber.size();
	vector <int> prefix(sizeInputNumber, 0);
	prefix_function(inputNumber, prefix, C);

	int violations = 0; // количество нарушений у владельца автомобиля

	ifstream fin(fileName);
	if (fin.is_open()) {
		string line;
		auto start = std::chrono::steady_clock::now();
		while (getline(fin, line)) {
			// Ищем номер автомобиля в текущей строке
			int n = line.length();
			int j = 0;
			for (int i = 0; i < n; i++) {
				while (j > 0 && line[i] != inputNumber[j]) {
					C++;
					j = prefix[j - 1];
				}
				C++;
				if (line[i] == inputNumber[j]) {
					j++;
				}
				C++;
				if (j == inputNumber.length()) {
					violations++;
					j = prefix[j - 1];
				}
			}
		}
		auto end = std::chrono::steady_clock::now();
		std::chrono::duration<double> elapsed_seconds = end - start;
		cout << "Количество нарушений у автомобиля с номером " <<
inputNumber << " - " << violations << endl
			<< "Время работы программы - " << elapsed_seconds.count()
<< endl
			<< "C = " << C << endl << endl;
	}
	else {
		cout << "Не удалось открыть файл" << endl << endl;
```

```
	}
}

void generateFile(int numberCar) {
	ofstream fout(fileName, ios::app);
	if (fout.is_open()) {
		char Auto[][8] = { "CITROEN", "fERRARI", "GENESIS", "PEUGEOT",
"RENAULT", "CHANGAN", "BUGATTI", "BENTLEY",
		"MAYBACH", "MCLAREN" };
		char number[][10] = {"A324TO799", "P308CC199", "X023TO109",
"H991KM199", "M555MM999", "T811BM238", "A753CB729",
		"H608TE899", "H888TT299", "O108TM877", "B101CC450", "M199MM199",
"P028AB308", "T909PP109", "T880AP709",
		"T111PT098", "A601HH023", "B221CC777", "A239BP078", "T237TP889",
"T303PE308", "O322PT230", "E443TT504" };
		for (int i = 0; i < numberCar; i++) {
			int index1 = rand() % 10;
			int index2 = rand() % 23;

			fout << "Автомобиль " << Auto[index1] << " с номером " <<
number[index2] << " попал в ДТП" << endl;
		}
		int size = 50 * numberCar;
		fout.close();
		cout << "Данные успешно записаны" << endl << "Количество
символов - " << size << endl << endl;
	}
	else {
		cout << "Не удалось открыть файл" << endl << endl;
	}
}

void coutFile() {
	cout << "Список машин, учавствовавших в ДТП:" << endl;
	ifstream fin(fileName);
	if (fin.is_open()) {
		string line;
		while (getline(fin, line)) {
			cout << line << endl;
		}
		fin.close();
		cout << endl;
	}
	else {
		cout << "Не удалось открыть файл" << endl << endl;
	}
}

int main() {
	setlocale(LC_ALL, "Rus");
	while (true){
		cout << "1 - Добавить машину" << endl
			<< "2 - Вывести все автомобили" << endl
			<< "3 - Найти автомобиль методом поиска Кнута-Мориса-
Пратта" << endl
			<< "0 - Завершить работу программы" << endl << endl
			<< "Выберите 1, 2, 3, 0: ";
		int choise, size = 0;
		cin >> choise;
		switch (choise)
		{
		case(1):
		{
			int carNumber;
```

```
                cout << "Сколько машин добавить?" << endl;
                cin >> carNumber;
                generateFile(carNumber);
                break;
            }
            case(2):
            {
                coutFile();
                break;
            }
            case(3):
            {
                string inputNumber;
                cout << "Введите номер автомобиля: ";
                cin >> inputNumber;
                long long int C = 0;
                coutNumberCar(inputNumber, C);
                break;
            }
            case(0):
                return 0;
                break;
            default:
                cout << "Вы ввели некорректное значение" << endl << endl;
                break;
            }
        }
}
```

Таблица 3. Код программы для задачи №1

```
#include <iostream>
#include <regex>
#include <string>

using namespace std;

bool isValidDate(string str) {
    regex date("^((0[1-9]|[12]\\d|3[01])\\.(0[1-9]|1[0-2])\\.([1-
9]\\d{3}))$");
    // шаблон даты: dd.mm.yyyy

    return regex_match(str, date);
}

int main() {
    setlocale(LC_ALL, "Rus");
    while (true) {
        string str;
        cout << endl << "0 - Завершить работу программы" << endl;
        cout << "Введите дату в формате \"dd.mm.yyyy\": ";
        getline(cin, str);
        if (str != "0"){
            if (isValidDate(str)) {
                cout << "Введенная дата является корректной датой с 1000
года.\n";
            }
            else {
                cout << "Введенная строка не является корректной датой.\n";
            }
        }
        else {
            return 0;
        }
```

```
        }
}
```

Таблица 4. Код программы для задачи №2