Отчёт по лабораторной работе №1. (В. 6)  
  
Задачи:  
 1. Чтение данных из файла в динамический массив.

2. Просмотр данных, хранящихся в массиве.

3. Корректировка данных заданной строки.

4. Вычисления.

5. Сохранение данных в новом файле (имя файла вводится с клавиатуры).

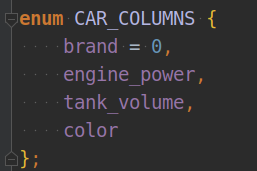
Для выполнения этой лабораторной работы были подключены 3 библиотеки:

#include <iostream> - для ввода-вывода

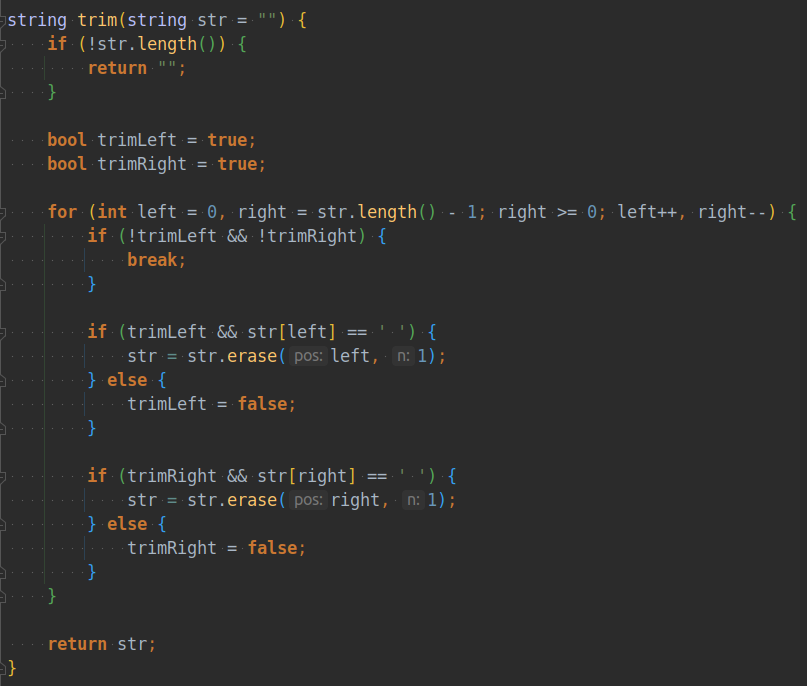
#include <fstream> - для работы с файлами

#include <string> - для удобной работы со строками

Так же было добавлено перечисление, для работы с колонками данных в файле



Была добавлена функция для удаления лишних пробелов между данными колонок



Для работы с файлом была создана структура FileTools, которое содержит в себе следующие атрибуты:

fstream file - файл, чьи данные будут использоваться в дальнейшем

**int** rowCount = 0 - количество строк в файле

**int** FILE\_ROW\_COLUMNS = 4 - количество колонок в строке файла

string\*\* fileData - данные из файла

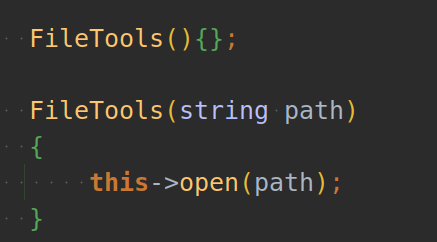
string path - путь до файла

**const int** END\_EDIT = 0 - флаг остановки редактирования файла

**const int** CONTINUE\_EDIT = 1 - флаг продолжения редактирования файла

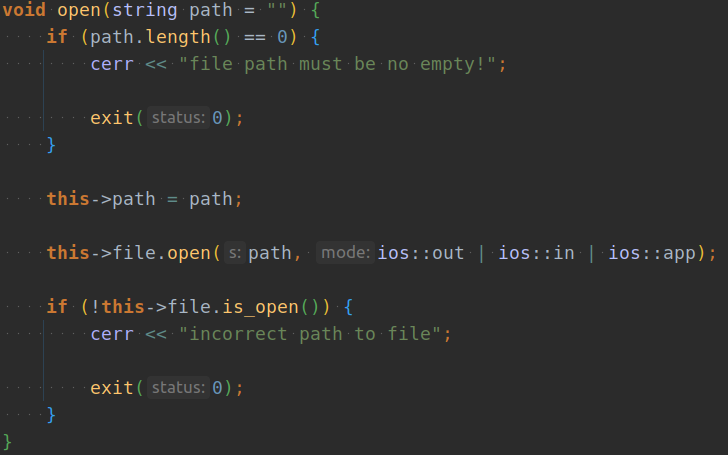
**const int** CHANGE\_EDIT\_ROW = 2 - флаг смены строки для редактирования

Для работы с этой структуры было создано два конструктора:

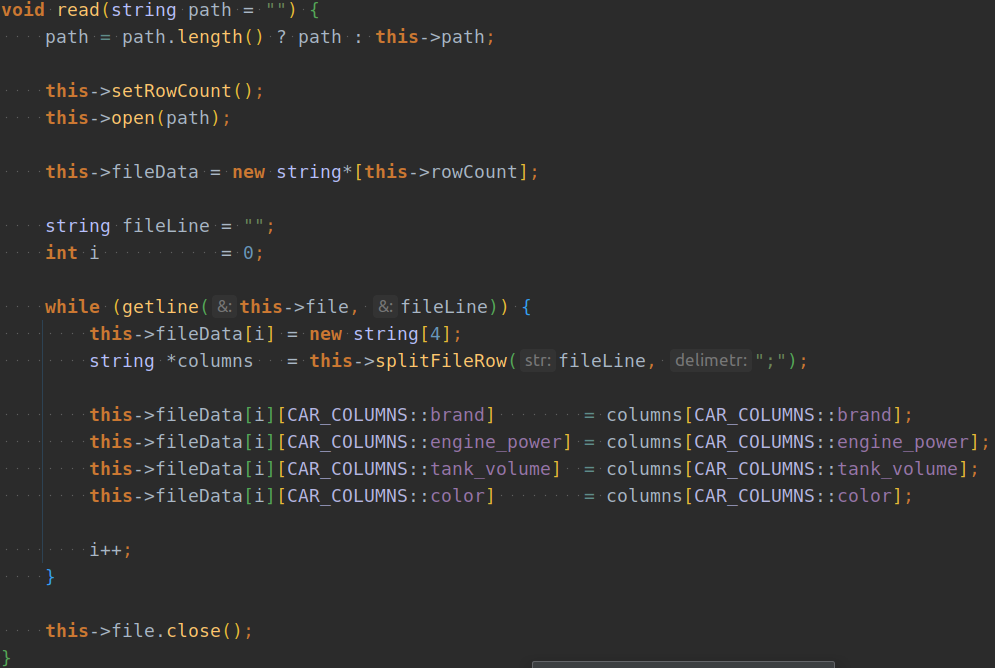


Второй конструктор принимает в качестве аргумента путь до файла

Для работы с файлом были созданы следующие методы:  
  
**void** open(string path = "") - метод открытия файла



**void** read(string path = "") - метод чтения данных из файла



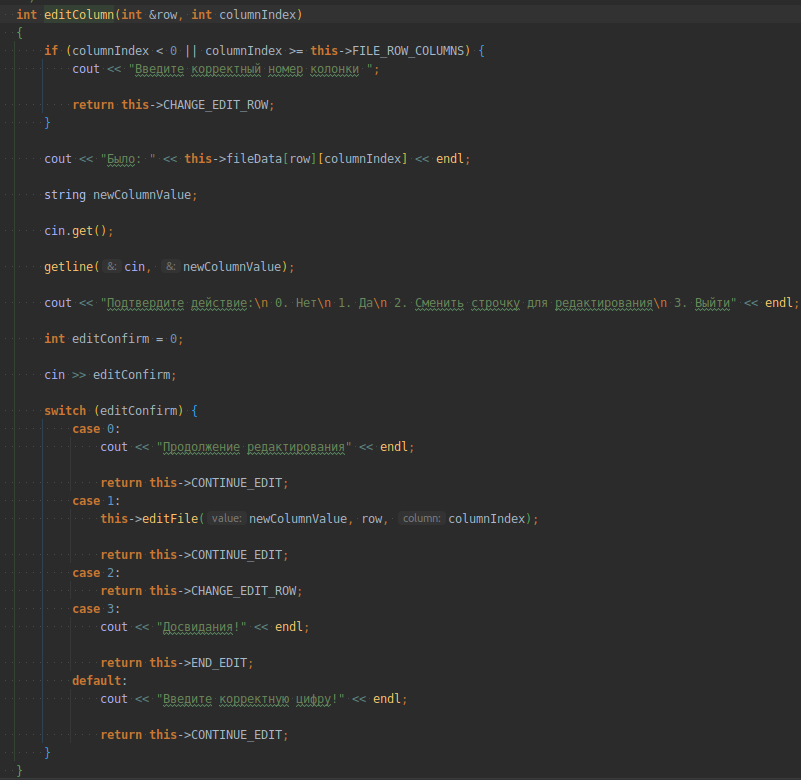
**void** add() - метод добавления данных в файл



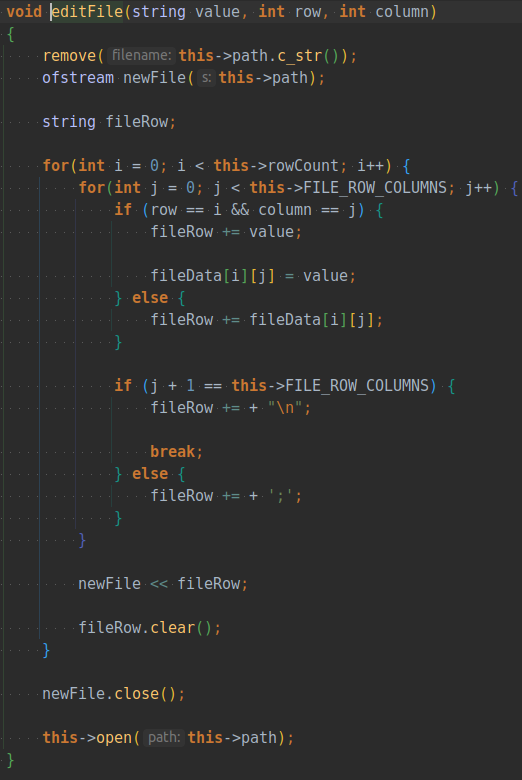
**void** editRow(**int** rowIndex = -1) - метод редактирования строки



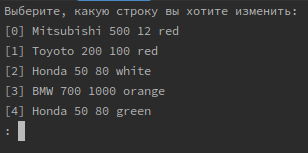
**int** editColumn(**int** &row, **int** columnIndex) - метод редактирования колонки



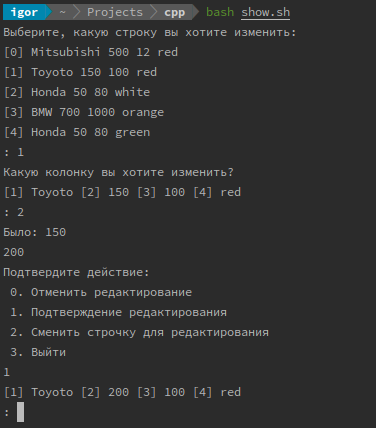
**void** editFile(string value, **int** row, **int** column) - метод редактирования файла



Эти методы выполняют пункты 1,3,5. Метод read читает данные из массива и записывает их в fileData, метод editRow (и вызываемые в этом методе editColumn и editFile) корректирует данные в заданной строке. Если вызывать editRow с rowIndex = - 1,то на выбор редактирования будут доступны все строки:



Выбрав одну из строк можно отредактировать одну из 4 колонок:

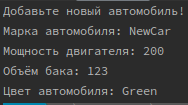


После выбора колонки для редактирования, вписывается число, которое заменит прошлое, после чего на выбор будет предложено 4 действия:

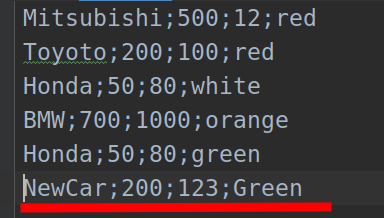
* Отменить изменения
* Подтвердить изменения
* Сменить строчку для редактирования
* Выйти из программы

Если отменить или подтвердить изменение, то для последующего редактирования будет выбрана та же строчка. Редактирование файла будет идти до тех пор, пока не нажать на “Выйти” или на комбинацию клавиш Ctrl+C

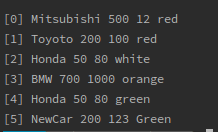
Вызвав метод add можно добавить новое значение в файл:



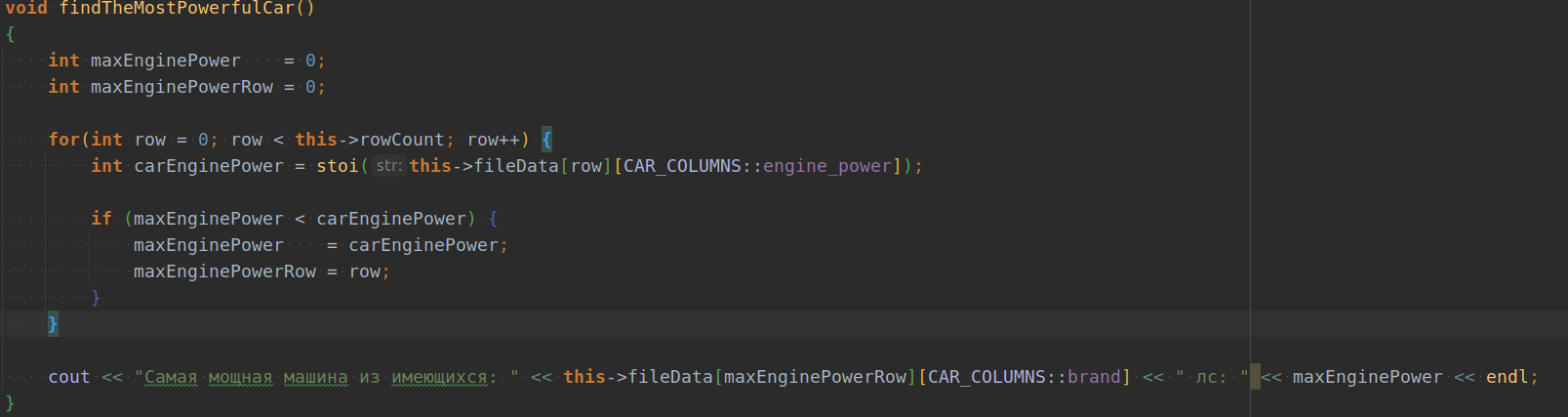
Результат выполнения:



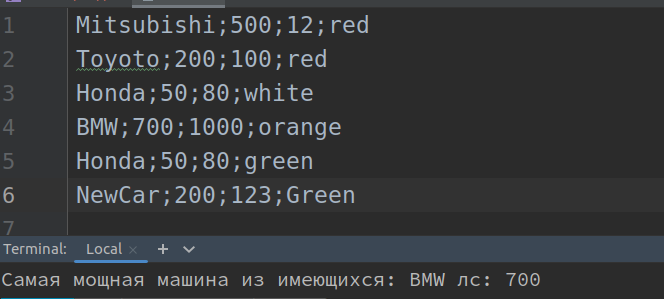
Для выполнения пункта 2 был реализован метод showData. Результат работы метода:



Для выполнения пункта 4 (Вычислить машину с самым мощным двигателем.) был реализован метод findTheMostPowerfulCar



Результат выполнения метода:



Пункт 5 выполняется при редактировании файла  
  
Весь код:  
  
#include <iostream>

#include <fstream>

#include <string>

**using namespace** std;

**enum** CAR\_COLUMNS {

*brand* = 0,

*engine\_power*,

*tank\_volume*,

*color*

};

string trim(string str = "") {

**if** (!str.length()) {

**return** "";

}

**bool** trimLeft = **true**;

**bool** trimRight = **true**;

**for** (**int** left = 0, right = str.length() - 1; right >= 0; left++, right--) {

**if** (!trimLeft && !trimRight) {

**break**;

}

**if** (trimLeft && str[left] == ' ') {

str = str.erase(left, 1);

} **else** {

trimLeft = **false**;

}

**if** (trimRight && str[right] == ' ') {

str = str.erase(right, 1);

} **else** {

trimRight = **false**;

}

}

**return** str;

}

/\*\*

\* Структура для работы с файлом

\*/

**struct** FileTools {

fstream file;

**int** rowCount = 0;

**int** FILE\_ROW\_COLUMNS = 4;

string\*\* fileData;

string path;

**const int** END\_EDIT = 0;

**const int** CONTINUE\_EDIT = 1;

**const int** CHANGE\_EDIT\_ROW = 2;

FileTools(){};

FileTools(string path)

{

**this**->open(path);

}

/\*\*

\* Метод открытия файла

\*

\* @param path

\*/

**void** open(string path = "") {

**if** (path.length() == 0) {

*cerr* << "file path must be no empty!";

exit(0);

}

**this**->path = path;

**this**->file.open(path, ios::out | ios::in | ios::app);

**if** (!**this**->file.is\_open()) {

*cerr* << "incorrect path to file";

exit(0);

}

}

/\*\*

\* Метод чтения файла

\*

\* @return

\*/

**void** read(string path = "") {

path = path.length() ? path : **this**->path;

**this**->setRowCount();

**this**->open(path);

**this**->fileData = **new** string\*[**this**->rowCount];

string fileLine = "";

**int** i = 0;

**while** (getline(**this**->file, fileLine)) {

**this**->fileData[i] = **new** string[4];

string \*columns = **this**->splitFileRow(fileLine, ";");

**this**->fileData[i][CAR\_COLUMNS::*brand*] = columns[CAR\_COLUMNS::*brand*];

**this**->fileData[i][CAR\_COLUMNS::*engine\_power*] = columns[CAR\_COLUMNS::*engine\_power*];

**this**->fileData[i][CAR\_COLUMNS::*tank\_volume*] = columns[CAR\_COLUMNS::*tank\_volume*];

**this**->fileData[i][CAR\_COLUMNS::*color*] = columns[CAR\_COLUMNS::*color*];

i++;

}

**this**->file.close();

}

/\*\*

\* Метод показа данных файла

\*/

**void** showData()

{

*cout* << endl;

**for**(**int** row = 0; row < **this**->rowCount; row++) {

*cout* << "[" << row << "] ";

**for**(**int** column = 0; column < **this**->FILE\_ROW\_COLUMNS; column++) {

*cout* << **this**->fileData[row][column] << " ";

}

*cout* << endl;

}

}

/\*\*

\* Метод добавления строки

\*/

**void** add() {

*cout* << "Добавьте новый автомобиль!\nМарка автомобиля: ";

string carBrand;

**this**->setValue(carBrand);

*cout* << "Мощность двигателя: ";

**int** enginePower;

**this**->setValue(enginePower);

*cout* << "Объём бака: ";

**int** tankVolume;

**this**->setValue(tankVolume);

*cout* << "Цвет автомобиля: ";

string color;

**this**->setValue(color);

**this**->open(**this**->path);

**this**->file << carBrand << ";" << enginePower << ";" << tankVolume << ";" << color << '\n';

**this**->file.close();

**this**->setRowCount();

}

/\*\*

\* Метод редактирования строки

\*

\* @param rowIndex

\*/

**void** editRow(**int** rowIndex = -1) {

**int** action = **this**->CONTINUE\_EDIT;

**while**(action) {

**if** (rowIndex == -1 || action == **this**->CHANGE\_EDIT\_ROW) {

*cout* << "Выберите, какую строку вы хотите изменить: ";

**this**->showData();

*cout* << ": ";

*cin* >> rowIndex;

}

**if** (rowIndex > **this**->rowCount) {

rowIndex = **this**->rowCount;

} **else if** (rowIndex < 0) {

*cerr* << "incorrect index";

exit(1);

}

**for**(**int** row = 0; row < **this**->rowCount; row++) {

**if** (row == rowIndex) {

*cout* << "Какую колонку вы хотите изменить?" << endl;

**while**(action) {

**for**(**int** column = 1; column <= **this**->FILE\_ROW\_COLUMNS; column++) {

*cout* << "[" << column << "] " << **this**->fileData[row][column-1] << " ";

}

*cout* << "\n: ";

**int** columnIndex = 0;

*cin* >> columnIndex;

action = **this**->editColumn(rowIndex, columnIndex-1);

**if** (action == **this**->CHANGE\_EDIT\_ROW) {

**break**;

}

}

**break**;

}

}

}

}

/\*\*

\* Метод редактирования колонки

\*

\* @param row

\* @param columnIndex

\*

\* @return

\*/

**int** editColumn(**int** &row, **int** columnIndex)

{

**if** (columnIndex < 0 || columnIndex >= **this**->FILE\_ROW\_COLUMNS) {

*cout* << "Введите корректный номер колонки ";

**return this**->CHANGE\_EDIT\_ROW;

}

*cout* << "Было: " << **this**->fileData[row][columnIndex] << endl;

string newColumnValue;

*cin*.get();

getline(*cin*, newColumnValue);

*cout* << "Подтвердите действие:\n 0. Отменить редактирование\n 1. Подтверждение редактирования\n 2. Сменить строчку для редактирования\n 3. Выйти" << endl;

**int** editConfirm = 0;

*cin* >> editConfirm;

**switch** (editConfirm) {

**case** 0:

*cout* << "Продолжение редактирования" << endl;

**return this**->CONTINUE\_EDIT;

**case** 1:

**this**->editFile(newColumnValue, row, columnIndex);

**return this**->CONTINUE\_EDIT;

**case** 2:

**return this**->CHANGE\_EDIT\_ROW;

**case** 3:

*cout* << "Досвидания!" << endl;

**return this**->END\_EDIT;

**default**:

*cout* << "Введите корректную цифру!" << endl;

**return this**->CONTINUE\_EDIT;

}

}

/\*\*

\* Непосредственное редактирование файла

\*

\* @param value

\* @param row

\* @param column

\*/

**void** editFile(string value, **int** row, **int** column)

{

remove(**this**->path.c\_str());

ofstream newFile(**this**->path);

string fileRow;

**for**(**int** i = 0; i < **this**->rowCount; i++) {

**for**(**int** j = 0; j < **this**->FILE\_ROW\_COLUMNS; j++) {

**if** (row == i && column == j) {

fileRow += value;

fileData[i][j] = value;

} **else** {

fileRow += fileData[i][j];

}

**if** (j + 1 == **this**->FILE\_ROW\_COLUMNS) {

fileRow += + "\n";

**break**;

} **else** {

fileRow += + ';';

}

}

newFile << fileRow;

fileRow.clear();

}

newFile.close();

**this**->open(**this**->path);

}

/\*\*

\* Метод для установки количества колонок в строке файла

\*

\* @param count

\*/

**void** setRowFileColumns(**int** count = 0) {

**if** (count <= 0) {

count = 4;

}

**this**->FILE\_ROW\_COLUMNS = count;

}

/\*\*

\* Метод разделения строки файла по делиметру

\* Пример: abc,dsf => ["abc","dsf"]

\*

\* @param str

\* @param delimetr

\* @return

\*/

string \*splitFileRow(string str = "", string delimetr = "")

{

**static** string splitedRow[4];

**int** start = 0;

**int** end = 0;

**for** (**int** i = 0; i < 4; i++) {

end = str.find(delimetr, end) + 1;

splitedRow[i] = trim(str.substr(start, end - start - 1));

start = end;

}

**return** splitedRow;

}

/\*\*

\* Метод нахождения самой мощной машины

\*/

**void** findTheMostPowerfulCar()

{

**int** maxEnginePower = 0;

**int** maxEnginePowerRow = 0;

**for**(**int** row = 0; row < **this**->rowCount; row++) {

**int** carEnginePower = stoi(**this**->fileData[row][CAR\_COLUMNS::*engine\_power*]);

**if** (maxEnginePower < carEnginePower) {

maxEnginePower = carEnginePower;

maxEnginePowerRow = row;

}

}

*cout* << "Самая мощная машина из имеющихся: " << **this**->fileData[maxEnginePowerRow][CAR\_COLUMNS::*brand*] << " лс: " << maxEnginePower << endl;

}

/\*\*

\* Метод установки количества строк в файле

\*/

**void** setRowCount()

{

string fileLine = "";

**int** i = 0;

**this**->file.close();

**this**->open(**this**->path);

**while** (getline(**this**->file, fileLine)) {

i++;

}

**this**->file.close();

**this**->rowCount = i;

}

/\*\*

\* Метод получения количества строк из файла

\*

\* @return

\*/

**int** getRowCount()

{

**return this**->rowCount;

}

/\*\*

\* Метод записи значения из консоли

\*

\* @param value

\*/

**void** setValue(string &value)

{

string buffer = "";

getline(*cin*, buffer);

value = buffer;

}

/\*\*

\* Метод записи значения из консоли

\*

\* @param value

\*/

**void** setValue(**int** &value)

{

string buffer = "";

getline(*cin*, buffer);

value = stoi(buffer);

}

};

**int** main() {

FileTools file;

file.open("./test.txt");

file.read();

file.findTheMostPowerfulCar();

**return** 0;

}