НИЖЕГОРОДСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ
УНИВЕРСИТЕТ ИМ. Р.Е. АЛЕКСЕЕВА

Кафедра "Вычислительные системы и технологии"

ПРОГРАММИРОВАНИЕ

Отчёт

по лабораторной работе № 4

Наследование, перегрузка операций

Вариант № 12

Выполнил студент группы 19-ИВТ-3
Сапожников Владислав Олегович
«___» _____ 20___ г.

Провел ст.преподаватель кафедры ВСТ Мартынов Д.С.
«___» _____ 20___ г.

Нижний Новгород 2020

Задание:

Согласно заданию, составить алгоритм и написать программу на языке C++. Программа компилируется и запускается под управлением ОС Linux.

В случае если студент разрабатывает программу, работающую под управлением операционной системы MS Windows, необходимо обеспечить максимальную переносимость кода. То есть не использовать не стандартные Microsoft-расширения языка *C++*, привязанные к среде разработки MS Visual Studio.

Разработанная программа должна содержать встроенную справочную информации, описывающую правила использования, цель назначения и информацию о разработчике. Аргументы запуска программа должна обрабатывать согласно рекомендациям POSIX.

Создать пользовательский класс TCharArray (массив), используемый для хранения элементов типа char. В данном классе должен быть реализован метод at для доступа к элементу символьного массива с проверкой корректности значения индекса элемента массива. Кроме того, необходимо перегрузить операцию [ ] для доступа к элементам массива. Используя класс TCharArray в качестве родительского, создать производный от него пользовательский класс String, используемый для хранения символьных строк. Для данного класса перегрузить следующие операции: ‘+’, ‘==’, ‘>’, ‘<’, ‘!=’

Разрабатываемая программа предназначена для хранения массива экземпляров класса. Перечень атрибут класса (членов-данных) определяется исходя их задания во второй лабораторной работе. Созданная программа должна поддерживать управление на уровне аргументов командной строки (аргументов запуска).

Поддерживаемые опции запуска:

--help либо *-h* - запуск программы в режиме получения справки. После вывода справочной информации программа завершает работу.

-c [*N*] [*file_name*] - запуск программы в режиме создания электронной таблицы записей, *N* – количество записей, *file_name* – имя бинарного файла, в котором будет сохранен массив (таблица) записей.

-r [*N*] [*file_name*] - запуск программы в режиме чтения содержимого бинарного файла *file_name*, на экран должны быть выведены не более *N* записей. Следует учесть, что реальное количество записей в файле может не совпадать с заданным значением *N*. Если заданный файл окажется пуст, либо по какой-либо причине программа не сможет его открыть, должно быть выдано соответствующее сообщение.

В случае, если программа будет запущена с неопределенными разработчиком аргументами, программа должна выдать соответствующее сообщение и вывести минимальную справку о корректных аргументах запуска. Это так же касается случая, когда программа запускается без аргументов.

Примечания:

В ходе выполнения лабораторной работы были написаны два класса:

1. MyString (динамическая строка типа char) более приближенная к классу std::string
2. MyString_stat (статическая строка) которая используется в классе Car
   Это необходимо для записи и чтения объектов, которые должны иметь определённый размер. При использовании динамической строки размер объекта не определен, либо это можно выполнить другими не известными мне способами.

Инструкция для запуска:

Для запуска введите:

. /main (<флаг_режима>) ([кол-во_записей]) (путь_к_текстовому_файлу)

Для дополнительной информации введите:

. /main [-h || --help]

Для проверки работы класса MyString(динамическая строка char) введите:

./main -strtest

Для создания таблицы (без указания кол-ва пунктов при запуске) введите:

. /main –c (<имя_файла>) - файл будет создан или перезаписан если файл с таким именем уже есть

Для создания таблицы с определённым кол-ом пунктов таблицы введите:

. /main –c –(<кол-во _пунктов>) (<имя_файла>) - файл будет создан или перезаписан если файл с таким именем уже есть

Для чтения полной таблицы из файла введите:

. /main –r (<имя_файла>)

Для чтение определенного кол-ва пунктов из таблицы введите:

. /main –r (<кол-во_пунктов>) (<имя_файла>)

Псевдокод (описание алгоритма работы программы):

Библиотека mystr

(содержит классы TCharArray, TCharArray_stat, MyString, MyString_stat, все методы функции, связанные с ними)

Заголовочный файл **mystr.h**

Начало заголовочного файла:

|

|Класс TCharArray:

||

||Защищенная секция:

|||str: указатель на тип char

|||SIZE: целочисленный беззнаковый

||Всё - защищенная секция:

||

||Публичная секция:

|||Конструктор TCharArray

|||Деструктор ~TCharArray
|||
|||метод at(int n) – возвращает ссылку на символьный тип, принимает целочисленную
||||переменную
|||
|||перегрузка оператора[] – доступ к элементу массива
||Всё - публичная секция:
||
|Всё – класс TCharArray:
|
|
|
|Класс MyString унаследованный от класса TCharArray:
||
||Публичная секция:
|||Конструктор MyString
|||
|||перегрузка оператора == относительно MyString
|||перегрузка оператора != относительно MyString
|||перегрузка оператора < относительно MyString
|||перегрузка оператора > относительно MyString
|||
|||
|||перегрузка оператора = относительно MyString
|||перегрузка оператора = относительно типа char
|||
|||перегрузка оператора + относительно MyString
|||
|||дружественный метод вывода объекта класса
|||
||Всё - публичная секция:
||
|Всё – класс MyString:
|
|
|
|MaxSIZE = 256: перечисляемый тип
|Класс TCharArray_stat:
||
||Защищенная секция:
|||str[MaxSIZE]: таблица символов
|||SIZE: целочисленный
||Всё - защищенная секция:
||
||Публичная секция:
|||Конструктор TCharArray
|||Деструктор ~TCharArray
|||

|||метод at(int n) – возвращает ссылку на символьный тип, принимает целочисленную
||||переменную
|||
|||перегрузка оператора[] – доступ к элементу массива
||Всё - публичная секция:
||
|Всё – класс TCharArray_stat
|
|
|
|Класс MyString_stat унаследованный от класса TCharArray_stat:
||
||Публичная секция:
|||Конструктор MyString_stat
|||
|||перегрузка оператора == относительно MyString_stat
|||перегрузка оператора != относительно MyString_stat
|||перегрузка оператора < относительно MyString_stat
|||перегрузка оператора > относительно MyString_stat
|||
|||перегрузка оператора = относительно MyString_stat
|||
|||дружественный метод вывода объекта класса
|||
|||статический метод alignLinesOfMyStr(MyString_stat &str, int width) – принимает ссылку
||||на объект MyString_stat, целочисленное значение width; возвращает символьную строку
|||
||Всё - публичная секция:
||
|Всё – класс MyString_stat
|
|
|
|Функция MyString_stat_check() – возвращает пустой тип
|
Конец заголовочного файла

Файл реализации **mystr.cpp**

Начало файла реализации:
|
|Реализация метода класса TCharArray::at(int i):
||Если (указатель str = нулевой указатель)
|||То: Вывод предупреждения
||||Выход из программы
||Всё – если
||
||Если (i<0 || i>= SIZE)
|||То: Вывод предупреждения

||||Выход из программы
||Всё – если
||
||Вернуть str[i]
||
|Всё - реализация метода
|
|
|
|перегрузка оператора [] класса TCharArray):
||Если (указатель str = нулевой указатель)
|||То: Вывод предупреждения
||||Выход из программы
||Всё – если
||
||Если (i<0 || i>= SIZE)
|||То: Вывод предупреждения
||||Выход из программы
||Всё – если
||
||Вернуть str[i]
||
|Всё - реализация метода
|
|
|
|реализация перегрузки оператора == (тождественно равно) класса MyString относительно
||класса MyString
||
||Если (размер данной строки <> размер переданной строки)
|||То: вернуть ложь
||Всё - если
||
||Цикл – для <i = 0; размер данной строки; 1>
|||Если (элемент[i] данной строки <> элемент[i] переданной строки)
||||То: вернуть ложь
|||Всё - если
||Всё – цикл для
||
||вернуть правда
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация перегрузки оператора != (тождественно неравно) класса MyString
||относительно класса MyString
||

||Если (размер данной строки <> размер переданной строки)
|||То: вернуть правда
||Всё - если
||
||Цикл – для <i = 0; размер данной строки; 1>
|||Если (элемент[i] данной строки <> элемент[i] переданной строки)
||||То: вернуть правда
|||Всё - если
||Всё – цикл для
||
||вернуть ложь
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация перегрузки оператора < (меньше) класса MyString относительно ||класса MyString
||Если (размер данный строки < размера переданной строки)
|||То: вернуть правда
||Всё - если
||
||Цикл – для <i = 0; размер данной строки; 1>
|||Если (элемент[i] данной строки < элемент[i] переданной строки)
||||То: вернуть правда
|||Всё - если
||Всё – цикл для
||
||вернуть ложь
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация перегрузки оператора > (больше) класса MyString относительно ||класса MyString
||Если (размер данный строки > размера переданной строки)
|||То: вернуть правда
||Всё - если
||
||Цикл – для <i = 0; размер данной строки; 1>
|||Если (элемент[i] данной строки > элемент[i] переданной строки)
||||То: вернуть правда
|||Всё - если
||Всё – цикл для
||
||вернуть ложь
||

|Всё – реализация перегрузки оператора
|
|
|
|реализация перегрузки оператора + (конкатенация) класса MyString относительно ||класса MyString
||
||newStr: объект класса MyString
||
||len1, len2, i = 0: целочисленные
||
||len1 = размер данной строки
||len2 = размер данной переданной строки
||поле SIZE = len1 + len2
||
||поле str объекта newStr = выделить динамическую память вплоть до SIZE+1
||
||Цикл – для < ; len1; 1>
|||элемент[i] объекта newStr = элемент[i] данной строки
||Всё – цикл для
||
||Цикл – для < i, j = 0; j< len2; 1;1>
|||элемент[i] объекта newStr = элемент[i] переданной строки
||Всё – цикл для
||
||последний элемент поля str объекта newStr = ‘терминирующий нуль’
||
||Вернуть объект newStr
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация перегрузки оператора = (присваивание) класса MyString относительно
||типа char
||
||Если (поле str данной строки <>)
|||То: удалить поле str данной строки
||Всё – если
||
||SIZE = размер переданного массива char
||поле str данной строки = выделить память под тип char вплоть до [SIZE+1]
||
||Цикл – для < i = 0 ; len1; 1>
|||элемент[i] данной строки = элемент[i] переданного массива
||Всё – цикл для
||
||последний элемент поля str объекта newStr = ‘терминирующий нуль’

||Вернуть данную строку
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация перегрузки оператора = (присваивание) класса MyString относительно
||класса MyString
||
||Если (поле str данной строки <>)
|||То: удалить поле str данной строки
||Всё – если
||
||SIZE = размер переданного строки класса MyString
||поле str данной строки = выделить память под тип char вплоть до [SIZE+1]
||
||Цикл – для < i = 0 ; len1; 1>
|||элемент[i] данной строки = элемент[i] строки класса MyString
||Всё – цикл для
||
||последний элемент поля str объекта newStr = ‘терминирующий нуль’
||Вернуть данную строку
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация дружественного оператора << (вывода) класса MyString относительно
||класса MyString
|
|
|
|Реализация метода класса TCharArray_stat::at(int i):
||Если (i<0 || i>= SIZE)
|||То: Вывод предупреждения
||||Выход из программы
||Всё – если
||
||Вернуть str[i]
||
|Всё - реализация метода
|
|
|
|перегрузка оператора [] класса TCharArray):
||Если (i<0 || i>= SIZE)
|||То: Вывод предупреждения
||||Выход из программы

||Всё – если
||
||Вернуть str[i]
||
|Всё - реализация метода
|
|
|
|реализация перегрузки оператора = (присваивание) класса MyString_stat относительно
||типа char
||
||скопировать в поле str данной строки содержимое поля str переданной строки
||
||Вернуть данную строку
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация дружественного оператора << (вывода) класса MyString_stat относительно
||класса MyString_stat
|
|
|
|реализация перегрузки оператора == (тождественно равно) класса MyString относительно
||класса MyString
||
||Если (размер данной строки <> размер переданной строки)
|||То: вернуть ложь
||Всё - если
||
||Цикл – для <i = 0; размер данной строки; 1>
|||Если (элемент[i] данной строки <> элемент[i] переданной строки)
||||То: вернуть ложь
|||Всё - если
||Всё – цикл для
||
||вернуть правда
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация перегрузки оператора != (тождественно неравно) класса MyString
||относительно класса MyString
||
||Если (размер данной строки <> размер переданной строки)
|||То: вернуть правда

||Всё - если
||
||Цикл – для <i = 0; размер данной строки; 1>
|||Если (элемент[i] данной строки <> элемент[i] переданной строки)
||||То: вернуть правда
|||Всё - если
||Всё – цикл для
||
||вернуть ложь
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация перегрузки оператора < (меньше) класса MyString относительно
||класса MyString
||Если (размер данный строки < размера переданной строки)
|||То: вернуть правда
||Всё - если
||
||Цикл – для <i = 0; размер данной строки; 1>
|||Если (элемент[i] данной строки < элемент[i] переданной строки)
||||То: вернуть правда
|||Всё - если
||Всё – цикл для
||
||вернуть ложь
||
|Всё – реализация перегрузки оператора
|
|
|
|реализация перегрузки оператора > (больше) класса MyString относительно
||класса MyString
||Если (размер данный строки > размера переданной строки)
|||То: вернуть правда
||Всё - если
||
||Цикл – для <i = 0; размер данной строки; 1>
|||Если (элемент[i] данной строки > элемент[i] переданной строки)
||||То: вернуть правда
|||Всё - если
||Всё – цикл для
||
||вернуть ложь
||
|Всё – реализация перегрузки оператора
|

|
|
|реализация функции MyString_check()
||предлагает ввести два символьных массива, затем создает две строки типа MyString и
||присваивает им значение введённых массивов. Затем по выбору из меню можно
||проверить все реализованные перегрузки для класса MyString
|
Конец заголовочного файла

Библиотека Car

(содержит класс Car, все его методы и так же функции, которые не являются членом класса, но работают с его объектами)

Заголовочный файл **car.h**

Начало заголовочного файла:
|
|Класс Car:
||
||Приватная секция:
|||
||||Описание полей класса:
||||| mark: строка библиотеки MyString_stat
||||| manufacturer: строка библиотеки MyString_stat
||||| yearOfIssue: целочисленный
||||| mileage: целочисленный
||||| ID: целочисленный беззнаковый
||||| price: целочисленный
||||Всё - описание
|||
||Публичная секция:
|||
|||Определение: Конструктор по умолчанию
|||
|||Определение: descriptor: статический беззнаковый целочисленный
|||
|||Определение: Метод для заполнения полей объекта - SetInfo
|||
|||Определение: Метод для вывода информации об объекте в консоль - PrintInfoConsole
||
|Всё - класс
|
|Определение: Функция для записи информации массива объектов в файл
|PrintInfoToFile(Принимаемые параметры: car – указатель на тип Car, sizeOfarray -
||целочисленный, file: ссылка на файловый поток записи)
|
|Определение: Функция для чтения информации обо всех объектах файла
|GetInfoFromFile(Принимаемые параметры: car – ссылка на тип Car, file: ссылка на
||файловый поток чтения)

|
|Перегрузка: Функция для чтения информации об определенном кол-ве объектов файла
|GetInfoFromFile(Принимаемые параметры: car – ссылка на тип Car, file: ссылка на
||файловый поток чтения, numberOfObj: целочисленный)
|
Конец заголовочного файла

Файл реализации **car.cpp**

Начало файла реализации:
|
|Инициализация descriptor (атрибут класса Car) = 0;
|
|Реализация Конструктора по умолчания класса Car
|| yearOfIssue = 0
|| mileage = 0
|| price = 0
|| descriptor += 1
|| ID = descriptor
|Всё - конструктор
|
|
|Метод класса Car для запыления информации об объекте
||Вывод: " Введите марку: "
||Ввод с проверкой и запись поля mark
||Вывод: " Введите изготовителя: "
||Ввод с проверкой и запись поля manufacturer
||Вывод: " Введите год выпуска: "
||Ввод с проверкой и запись поля yearOfIssue
||Вывод: " Введите пробег: "
||Ввод с проверкой и запись поля mileage
||Вывод: " Введите цену: "
||Ввод с проверкой и запись поля price
|Всё – метод
|
|
|
|Реализация метода PrintInfoConsole класса Car
||Вывод номера Записи
||Вывод столбца о Марке автомобиля
||Вывод столбца о Годе выпуска автомобиля
||Вывод столбца о Пробеге автомобиля
||Вывод столбца об Изготовителе автомобиля
||Вывод столбца о Цене автомобиля
||Вывод столбца о ID автомобиля
|Всё - метод
|
|
|Реализация функции PrintInfoToFile

||Цикл -для <I = 0, sizeOfarray, 1>
|||Чтение в поток записи file инф-ию об объекте Car
||Всё-цикл для
|
|
|Реализация функции GetInfoFromFile
||endposition, counter, position: целочисленный
||
||Установить указатель в конец файла
||endposition = позиция указателя в файле
||counter = endposition / размер объекта car
||
||Если (counter = 0)
|||To: Вывод "Читаемый файл пуст!!!"
||||Досрочный выход из программы
||Всё - если
||
||
||Вывод: "Если во время вывода таблица съехала, то перезаполните таблицу по размеру."
||Вывод: "Всего автомобилей в таблице: " counter
||
||Цикл – для <i = 0; counter; 1>
|||position = i*размер объекта Car
|||Установить указатель внутри файла на позицию - position
|||Чтение информации из файла в объект в Car
|||Вызов метода PrintInfoConsole для объекта Car
||Всё – цикл для
||
|Всё – функция GetInfoFromFile
|
|Перегрузка функции GetInfoFromFile
||endposition, counter, position: целочисленный
||
||Установить указатель в конец файла
||endposition = позиция указателя в файле
||counter = endposition / размер объекта car
||
||Если (counter = 0)
|||To: Вывод "Читаемый файл пуст!!!"
||||Досрочный выход из программы
||Всё - если
||
||
||Если (numberOfObj > counter)
|||To: "В файле нет указанного кол-ва пунктов!"
||||Досрочный выход из программы
||Всё – если
||

||
||Если (numberOfObj < 1)
|||To: "Введено неверное кол-во пунктов для чтения! "
||||Досрочный выход из программы
||Всё-если
||
||
||Вывод: "Если во время вывода таблица съехала, то перезаполните таблицу по размеру."
||Вывод: "Всего автомобилей в таблице: " counter
||
||Цикл – для <i = 0; numberOfObj; 1>
|||position = i*размер объекта Car
|||Установить указатель внутри файла на позицию - position
|||Чтение информации из файла в объект в Car
|||Вызов метода PrintInfoConsole для объекта Car
||Всё – цикл для
||
|Всё - перегрузка
|
Конец файла реализации

Библиотека heading

(содержит функции проверки открытия файлов, получение кол-во пунктов таблицы, проверки строки на содержание только цифр и функция выравнивания строки по заданной ширине)

Заголовочный файл **heading.h**

Начало заголовочного файла:
|
|подключение библиотек
|
|
|Определении функции для проверки строки на содержание только цифр
||CheckInputNumbers(Принимаемые параметры: str - ссылка на динамическую безразмерную ||символьную таблицу. Возвращает: целочисленное значение)
|
|
|Определение функции для выравнивания строки по заданной ширине
||alignLinesOfChar(Принимаемые параметры: str – символьная строка, width: ||целочисленный. Возвращает символьную строку)
|
|
|Определение функции для ввода кол-ва пунктов таблицы
||GetQuantityOfPoints(Возвращает целочисленное значение)
|
|Перегрузка GetQuantityOfPoints(Принимаемые параметры: argv – указатель на ||символьную Таблицу)
|

|
|Определение функции для проверки файлового потока для записи
||CheckFile(Принимаемые параметры: file – ссылка на файловый поток записи)
|
|Перегрузка функции для проверки файлового потока для чтения
||CheckFile(Принимаемые параметры: file – ссылка на файловый поток чтения)|
|
Конец заголовочного файла

Файл реализации **heading.cpp**

Начало файла реализации:
|
|Реализация функции CheckFile
||Если (file (файловый поток для записи) не открыт)
|||То: Вывод "Произошла ошибка при открытие файлового потока для записи!"
||||Досрочный выход из файла
||Всё - если
|Всё – функция CheckFile
|
|Перегрузка функции CheckFile
||Если (file (файловый поток для чтения) не открыт)
|||То: Вывод "Произошла ошибка при открытие файлового потока для чтения!"
||||Досрочный выход из файла
||Всё - если
|Всё – перегрузка CheckFile
|
|
|Реализация функции для проверки строки на содержание только цифр
||CheckInputNumbers(str – указатель на символьную таблицу)
||len: целочисленный
||
||len = длина символьной таблицы str
||Цикл – для <i = 0, len, 1>
|||Если (str[i] < '0' ИЛИ str[i] > '9')
||||То: Вернуть 0
|||Всё – если
||Всё – цикл для
||
||Вернуть 1
||
|Всё функция CheckInputNumbers
|
|
|Реализация функции GetQuantityOfPoints
||numberOfLines: символьная таблица [12]
||NumbLines: целочисленный
||

||Вывод: "Был выбран режим без указания кол-ва пунктов в таблице при запуске
|||программы."
||
||Пока (правда)
|||Вывод: "Укажите кол-во пунктов в таблице: "
|||Если (Ввод корректный И numberOfLines > 0 И CheckInputNumbers(numberOfLines))
||||Если (numberOfLines > 100)
|||||Вывод: "Превышен лимит записей!"
|||||Досрочный выход из программы
||||Всё - если
||||NumbLines = numberOfLines
||||прервать
|||Всё - если
|||Вывод: "Неправильный ввод! Введите корректное число пунктов таблицы."
||Вернуть NumbLines
|
|Всё – функция GetQuantityOfPoints
|
|
|Перегрузка функции GetQuantityOfPoints
||numberOfLines: целочисленный
||
||Если (argv < 1 ИЛИ !CheckInputNumbers(argv))
|||Вывод: "Было задано неверное кол-во пунктов при запуске!"
|||Досрочный выход из программы
||Всё – если
||
||numberOfLines = argv
||Вернуть numberOfLines
|Всё – перегрузка GetQuantityOfPoints
|
|
|Реализация функции alignLinesOfChar
||len, diff, pad1, pad2: целочисленный
||
||Если (len > width)
|||То: Вернуть str
||Всё – если
||
||diff = width -len
||pad1 = diff/2
||pad2 = diff - pad1
||
||Вернуть символьная строка из pad1 пробелов + str + символьная строка из pad2
||
|Конец функции alignLinesOfChar
|
Конец файла реализации

Функция main () – вход в программу

Функция main: -точка входа в программу:

Принимаемые параметры:
- argc: целочисленный (отвечает за кол-во передаваемых параметров при запуске)
- Таблица argv: символьный (отвечает за содержимое передаваемых параметров при запуске)

Начало:

|
|**Множественный выбор** (argc)
||
||*argc = 2*
|||
|||Если (встречен флаг "-h" или "--help")
||||То: Вызов функции PrintFullReferens (вывод полной справки о программе)
|||||Выход из программы
||||Иначе:
|||||Вызов функции PrintShortreferens (вывод короткой справки о программе)
|||||Выход из программы
|||Всё-если
||
||
||
|||Если (встречен флаг -"strtest")
||||То: Вызов функции MyString_check() (проверка функционала класса MyString)
|||Всё-если
||
||
||
||*argc = 3*
|||
|||
|||Если (встречен флаг "-c")
||||numberOfLines: целочисленный
||||arrayOfCar: Динамическая таблица[numberOfLines]
||||То: открыть файловый поток для записи – res
|||||Вызов функции CheckFile – передать в функцию: res
|||||
|||||Вызов функции PrintLogotip
|||||Вызов функции PrintHat – передать: 3,12
|||||
|||||numberOfLines ← результат функции GetQuantityOfPoints
|||||
|||||Вызов функции PrintWarning

|||||
|||||Цикл-для <i = 0, numberOfLines, 1>
|||||arrayOfCar[i] ← метод SetInfo
|||||Записать в file информацию об объекте arrayOfCar[i]
|||||Всё – цикл для
|||||Удалить динамическую таблицу arrayOfCar
|||||
|||||Закрытие файлового потока для записи res
|||||Вывод: "Завершение работы программы!"
|||||Прервать
|||||
|||Всё – если (встречен флаг "-c"))
|||
|||
|||Если (встречен флаг "-r")
||||
||||Создание объекта car класса Car
||||
||||Открыть поток text (путь (argv[3])) для чтения
||||Вызов функции CheckFile – передать в функцию: text
||||
||||Вызов функции GetInfoFromFile ← car, text
||||
||||Закрытие файлового потока для чтения text
||||
||||Вывод: "Завершение работы программы!"
||||Прервать
||||
|||Всё – если (встречен флаг "-r")
|||
|||Вызов функции PrintErrorFlag
|||Прервать
||
||
||*argc = 4*
|||
|||Если (встречен флаг "-c")
||||numberOfLines: целочисленный
||||arrayOfCar: Динамическая таблица[numberOfLines]
||||To: открыть файловый поток для записи – res
|||||Вызов функции CheckFile – передать в функцию: res
|||||
|||||Вызов функции PrintLogotip
|||||Вызов функции PrintHat – передать: 3,12
|||||
|||||numberOfLines ← результат функции GetQuantityOfPoints ← argv[2]
|||||
|||||Цикл-для <i = 0, numberOfLines, 1>

||||||arrayOfCar[i] ← метод SetInfo
||||||Записать в file информацию об объекте arrayOfCar[i]
|||||Всё – цикл для
|||||
|||||
|||||Закрытие файлового потока для записи res
|||||Вывод: "Завершение работы программы!"
|||||Прервать
||||
|||Всё – если (встречен флаг "-c"))
|||
|||
|||Если (встречен флаг "-r")
||||
||||numberOfLines: целочисленный
||||Создание объекта car класса Car
||||
||||To: Открыть поток text (путь (argv[3])) для чтения
|||||Вызов функции CheckFile – передать в функцию: text
|||||
|||||numberOfLines ← результат функции GetQuantityOfPoints ← argv[2]
|||||
|||||Вызов функции GetInfoFromFile ← car, text, numberOfLines
|||||
|||||Закрытие файлового потока для чтения text
|||||
||||Вывод: "Завершение работы программы!"
||||Прервать
||||
|||Всё – если (встречен флаг "-r")
|||
|||
|||Вызов функции PrintErrorFlag
|||Прервать
||
||
||*По умолчанию:*
|||
|||Вызов функции PrintShortreferens (вывод короткой справки о программе)
|||Досрочный выход из программы
|||Прервать
||
||Всё - Множественный выбор (argc)
|
|Вернуть 0
|
Конец

Текст программы:
Библиотека mystr
Заголовочный файл **mystr.h**

```
#pragma once
#include "heading.h"

////////////////////////////////////////////////////////////////////////////////////////////
//                         Заголовочный файл библиотеки mystr                             //
//                           Лабораторной работы №4                                       //
//                                 Вариант №12                                            //
//                     Разработчик: студент группы 19-ИВТ-3                               //
//                             Сапожников Владислав                                       //
////////////////////////////////////////////////////////////////////////////////////////////



////////////////////////////////////////////////////////////////////////////////////////////
//     Объявление методов и конструкторов класса TCharArray (Динамическая строка)        //
////////////////////////////////////////////////////////////////////////////////////////////
class TCharArray{
 protected  char *str;
  unsigned int SIZE;

 public:
  TCharArray();
  ~TCharArray();

  char& at(int n);
  char& operator[](int n);
};



////////////////////////////////////////////////////////////////////////////////////////////
//      Объявление методов и конструкторов класса MyString (Динамическая строка)         //
////////////////////////////////////////////////////////////////////////////////////////////
class MyString : public TCharArray{
 public:
  MyString();

  bool operator == (MyString &str) const;
  bool operator != (MyString &str) const;
  bool operator <  (MyString &str) const;
  bool operator >  (MyString &str) const;

  MyString operator  + (MyString &str);

  MyString& operator = (const char* otherStr);

  MyString& operator = (const MyString& otherStr);


  friend std::ostream& operator << (std::ostream& os, const MyString& str);
};
```

```
////////////////////////////////////////////////////////////////////////////////////////
//     Объявление методов и конструкторов класса TCharArray_stat (Статическая строка)       //
////////////////////////////////////////////////////////////////////////////////////////
enum{MaxSIZE = 256};
class TCharArray_stat{
 protected:
  char str[MaxSIZE];
  int SIZE;

 public:
  TCharArray_stat();

  char& at(int n);
  char& operator[](int n);
};



////////////////////////////////////////////////////////////////////////////////////////
//       Объявление методов и конструкторов класса MyString_stat (Статическая строка)       //
////////////////////////////////////////////////////////////////////////////////////////
class MyString_stat : public TCharArray_stat{

 public:
  MyString_stat();

  MyString_stat& operator = (const char* otherStr);
  static std::string alignLinesOfMyStr(MyString_stat &str, int width);



  friend std::ostream& operator << (std::ostream& os, const MyString_stat& str);



  bool operator == (MyString_stat &str) const;
  bool operator != (MyString_stat &str) const;
  bool operator <  (MyString_stat &str) const;
  bool operator >  (MyString_stat &str) const;
};

void MyString_check();
```

Файл реализации **mystr.cpp**

```
#include "mystr.h"

////////////////////////////////////////////////////////////////////////////////////////
//                            Файл реализации библиотеки mystr                            //
//                                Лабораторной работы №4                                  //
//                                     Вариант №12                                        //
//                          Разработчик: студент группы 19-ИВТ-3                          //
//                                Сапожников Владислав                                    //
////////////////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////////////////
//      Реализация методов и конструкторов класса TCharArray(динамическая строка)          //
////////////////////////////////////////////////////////////////////////////////////////

//                        Конструктор по умолчанию TCharArray                            //
TCharArray::TCharArray():str(nullptr), SIZE(0){};



//                               Деструктор TCharArray                                   //
TCharArray::~TCharArray(){
    if(this->str != nullptr){
        delete[] this->str;
        str = nullptr;
    }
}



//               метод at - безопасный доступ к элементу массива                         //
char& TCharArray::at(int i){
    if(this->str == nullptr){
        cerr << endl << red << " Ошибочный индекс: " << i  << "! Выход за пределы массива!!!"
<< endl;
        cerr << " Данная строка нулевой длины и не содержит символов!" << endl;
        cerr << " Исправте ошибку и попробуйте еще раз!" << reset << endl << endl;
        exit(1);
    }
    if(i<0 || i>= SIZE){
        cerr << red << endl << "Ошибочный индекс: "  << i << "! Выход за пределы массива!!!"
<< endl;
        cout << "Исправте ошибку и попробуйте еще раз!" << reset << endl << endl;
        exit(1);
    }
    return str[i];
}



//             перегруженный оператор[], тоже с проверкой на безопасность                 //
char& TCharArray::operator[](int i){
     if(this->str == nullptr){
        cerr << endl << red << " Ошибочный индекс: " << i  << "! Выход за пределы массива!!!"
<< endl;
        cerr << " Данная строка нулевой длины и не содержит символов!" << endl;
        cerr << " Исправте ошибку и попробуйте еще раз!" << reset << endl << endl;
        exit(1);
     }
     if(i<0 || i>= SIZE){
        cerr << red << endl << "Ошибочный индекс: "  << i << "! Выход за пределы массива!!!"
<< endl;
        cout << "Исправте ошибку и попробуйте еще раз!" << reset << endl << endl;
        exit(1);
     }

     return str[i];
}
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////////////////
//       Реализация методов и конструкторов класса MyString (динамическая строка)    //
////////////////////////////////////////////////////////////////////////////////////////

//                          Конструктор по умолчанию MyString                         //
MyString::MyString():TCharArray(){};



//                        Оператор тождественно равно MyString                        //
bool MyString::operator == (MyString &OtherStr) const {
    if(this->SIZE != OtherStr.SIZE){
        return  false;
    }
    for(int i = 0; i < this->SIZE; i++){
        if(this->str[i] != OtherStr.str[i]){
            return false;
        }
    }
    return true;
};



//                      Оператор тождественно не равно MyString                       //
bool MyString::operator != (MyString &OtherStr) const {
    if(this->SIZE != OtherStr.SIZE){
        return true;
    }
    for(int i=0; i < this->SIZE; i++){
        if(this->str[i] != OtherStr.str[i]){
            return true;
        }
    }
    return false;
};



//                             Оператор меньше MyString                               //
bool MyString::operator < (MyString &OtherStr) const {
    if(this->SIZE < OtherStr.SIZE){
        return true;
    }
    for(int i=0; i < this->SIZE; i++){
        if(this->str[i] < OtherStr.str[i]){
            return true;
        }
    }
    return false;
};



//                              Оператор больше MyString                              //
bool MyString::operator > (MyString &OtherStr) const {
    if(strlen(this->str) > strlen(OtherStr.str)){
        return true;
    }
    for(int i=0; i < this->SIZE; i++){
```

```
        if(this->str[i] > OtherStr.str[i]){
            return true;
        }
    }
    return false;
};



//                              конкатенация строк MyString                                //
MyString MyString::operator + (MyString &OtherStr) {
    MyString newStr;

    unsigned int len1 = strlen(this->str);
    unsigned int len2 = strlen(OtherStr.str);
    SIZE = len1+len2;

    newStr.str = new char[SIZE+1];

    int i=0;

    for(       ; i < len1; i++){
        newStr.str[i] = this->str[i];
    }
    for(int j=0; j < len2; i++, j++){
        newStr.str[i] = OtherStr.str[j];
    }
    newStr.str[SIZE] = '\0';
    return newStr;
}



//                              оператор присваивания MyString                             //
MyString& MyString::operator =(const char* otherStr){
    if(str != nullptr){
        delete[] this->str;
    }

    SIZE = strlen(otherStr);
    this->str = new char [SIZE+1];

    for(int i = 0; i < SIZE; i++){
        this->str[i] = otherStr[i];
    }

    this->str[SIZE] = '\0';

    return *this;
}



//                               оператор присваивания MyString                             //
MyString& MyString::operator =(const MyString& otherStr){
    if(str != nullptr){
        delete[] this->str;
    }

    SIZE = strlen(otherStr.str);
    this->str = new char [SIZE+1];
```

```
    for(int i = 0; i < SIZE; i++){
        this->str[i] = otherStr.str[i];
    }

    this->str[SIZE] = '\0';
    return *this;
}



//                      Дружественный метод вывода MyString                                    //
std::ostream& operator << (std::ostream& os, const MyString& OtherStr){
    int allStr = strlen(OtherStr.str);
    for(int i = 0; i < allStr; i++){
            os  << OtherStr.str[i];
        }
    return os;
};
////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////////



////////////////////////////////////////////////////////////////////////////////////////////////
//     Реализация методов и конструкторов класса TCharArray_stat(статическая строка)        //
////////////////////////////////////////////////////////////////////////////////////////////////

//                     Конструктор по умолчанию TCharArray_stat                               //
TCharArray_stat::TCharArray_stat(){
    strcpy(str,"");
    SIZE = strlen(this->str);
};



//                                    метод at()                                              //
char& TCharArray_stat::at(int i){
    if(i<0 || i>= SIZE){
        cerr << red << endl << "Ошибочный индекс: "  << i << "! Выход за пределы массива!!!"
<< endl;
        cout << "Исправте ошибку и попробуйте еще раз!" << reset << endl << endl;
        exit(1);
    }
    return str[i];
}



//                перегруженный оператор[], тоже с проверкой на безопасность               //
char& TCharArray_stat::operator[](int i){
     if(i<0 || i>= SIZE){
         std::cout  <<  std::endl  <<  "  Ошибочный  индекс:  "  <<  i  <<  "!  Выход  за  пределы
массива!!!" << std::endl;
         std::cout << " Исправте ошибку и попробуйте еще раз!" << std::endl << std::endl;
         exit(1);
     }

     return str[i];
}
////////////////////////////////////////////////////////////////////////////////////////////////
```

```
////////////////////////////////////////////////////////////////////////////////////////////
////////////////////////////////////////////////////////////////////////////////////////////



////////////////////////////////////////////////////////////////////////////////////////////
//        Реализация методов и конструкторов класса MyString_stat(статическая строка)       //
////////////////////////////////////////////////////////////////////////////////////////////



//                         Конструктор по умолчанию MyString_stat                           //
MyString_stat::MyString_stat():TCharArray_stat(){};



//                          оператор присваивания MyString_stat                             //
MyString_stat& MyString_stat::operator =(const char* otherStr){
    strcpy(this->str, otherStr);
    return *this;
}



//                    Дружественный метод вывода MyString_stat                              //
std::ostream& operator << (std::ostream& os, const MyString_stat& OtherStr){
    int allStr = strlen(OtherStr.str);
    for(int i = 0; i < allStr; i++){
            os  << OtherStr.str[i];
        }
    return os;
};



//                     Оператор тождественно равно MyString_stat                           //
bool MyString_stat::operator == (MyString_stat &OtherStr) const {
    if(this->SIZE != OtherStr.SIZE){
        return  false;
    }
    for(int i = 0; i < this->SIZE; i++){
        if(this->str[i] != OtherStr.str[i]){
            return false;
        }
    }
    return true;
};



//                   Оператор тождественно не равно MyString_stat                          //
bool MyString_stat::operator != (MyString_stat &OtherStr) const {
    if(this->SIZE != OtherStr.SIZE){
        return true;
    }
    for(int i=0; i < this->SIZE; i++){
        if(this->str[i] != OtherStr.str[i]){
            return true;
        }
    }
    return false;
```

```
};


//                          Оператор меньше MyString_stat                                   //
bool MyString_stat::operator < (MyString_stat &OtherStr) const {
    if(this->SIZE < OtherStr.SIZE){
        return true;
    }
    for(int i=0; i < this->SIZE; i++){
        if(this->str[i] < OtherStr.str[i]){
            return true;
        }
    }
    return false;
};



//                          Оператор больше MyString_stat                                   //
bool MyString_stat::operator > (MyString_stat &OtherStr) const {
    if(strlen(this->str) > strlen(OtherStr.str)){
        return true;
    }
    for(int i=0; i < this->SIZE; i++){
        if(this->str[i] > OtherStr.str[i]){
            return true;
        }
    }
    return false;
};



//                          Функция проверки класса MyString_stat                           //
void MyString_check(){
        char buff[256];

        cout << endl << "\tКласс MyString - динамическая строка char" << endl;

        cout << endl << red << "Warning!!!" << reset << " рекомендуемый язык для ввода: " <<
yellow << "English" << reset << endl << endl;

        cout << "Введите содержимое 1ой строки: ";
        cin.get(buff,256);
        cin.ignore(256, '\n');
        MyString str1;
        str1 = buff;

        cout << "Введите содержимое 2ой строки: ";
        cin.get(buff,256);
        cin.ignore(256, '\n');
        MyString str2;
        str2 = buff;

        cout << "1ая строка: " << endl << "  " << str1 << endl << endl;
        cout << "2ая строка: " << endl << "  " << str2 << endl << endl;

        cout << "Выберите оператор для проверки: " << std::endl;
        cout << "\t1 - \"==\"" << std::endl;
        cout << "\t2 - \"!=\"" << std::endl;
```

```
 cout << "\t3 - \">\"" << std::endl;
 cout << "\t4 - \"<\"" << std::endl;
 cout << "\t5 - \"+\"" << std::endl;
 cout << "\t6 - \"at()\"" << std::endl;
 cout << "\t7 - \"[]\"" << std::endl;
 cout << "Ваш выбор: ";

int answer;

std::cin >> answer;

switch(answer){
 case 1:{
     if(str1 == str2){
         std::cout <<" строка №1 равна строке №2" << std::endl;
     }
     else{
         std::cout <<" строка №1 не равна строке №2" << std::endl;
     }
     break;
 }

 case 2:{
     if(str1 != str2){
         std::cout <<" строка №1 не равна строке №2" << std::endl;
     }
     else{
         std::cout <<" строка №1 равна строке №2" << std::endl;
     }
     break;
 }

 case 3:{
     if(str1 > str2){
         std::cout <<" строка №1 больше строки №2" << std::endl;
     }
     else{
         std::cout <<" строка №1 не больше строки №2" << std::endl;
     }
     break;
 }
 case 4:{
     if(str1 < str2){
         std::cout <<" строка №1 меньше строки №2" << std::endl;
     }
     else{
         std::cout <<" строка №1 не меньше строки №2" << std::endl;
     }
     break;
 }
 case 5:{
     MyString str3;
     str3 = str1 + str2;
     std::cout << " Результирующая строка:" << std::endl;
     std::cout << str3 << std::endl;
     break;
 }
 case 6:{
     std::cout << " Выберите номер строки с которой хотите работать: ";
     int m; std::cin >> m;
     if(m < 1 || m > 2 ){
```

```
                cerr << red << "Выбрана неверная строка!" << reset << endl;
                exit(0);
            }
            switch (m)
            {
            case 1:{
                std::cout << " Введите номер элемента к которому хотите получить доступ(начало
отсчета индексов с 1): " << std::endl;
                int n;
                std::cin >> n;
                n -= 1;
                std::cout << "Выбранный элемент: "  << str1.at(n) << std::endl;
                break;
            }
            case 2:{
                std::cout << " Введите номер элемента к которому хотите получить доступ(начало
отсчета индексов с 1): " << std::endl;
                int n;
                std::cin >> n;
                n -= 1;
                std::cout << "Выбранный элемент: "  << str2.at(n) << std::endl;
                break;
            }

            }
            break;
        }

        case 7:{
            std::cout << " Выберите номер строки с которой хотите работать: ";
            int m; std::cin >> m;
            if(m < 1 || m > 2 ){
                cerr << red << "Выбрана неверная строка!" << reset << endl;
                exit(0);
            }
            switch (m)
            {
            case 1:{
                std::cout << " Введите номер элемента к которому хотите получить доступ(начало
отсчета индексов с 1): " << std::endl;
                int n;
                std::cin >> n;
                n -= 1;
                std::cout << "Выбранный элемент: "  << str1[n] << std::endl;
                break;
            }
            case 2:{
                std::cout << " Введите номер элемента к которому хотите получить доступ(начало
отсчета индексов с 1): " << std::endl;
                int n;
                std::cin >> n;
                n -= 1;
                std::cout << "Выбранный элемент: "  << str2[n] << std::endl;
                break;
            }

            }
            break;
        }
    }
}
```

```
//                статическая функция для выравнивания строк MyString_stat                //
std::string MyString_stat::alignLinesOfMyStr(MyString_stat &Mystr, int width){
    std::string str = Mystr.str;

    int len = str.length();
    if(len > width){return str;};

    int diff = width -len;
    int pad1 = diff/2;
    int pad2 = diff - pad1;

    return std::string(pad1,' ') + str + std::string(pad2,' ');
};
```

Библиотека Car

Заголовочный файл **car.h**

```
#ifndef _CAR_H_
#define _CAR_H_
#include "heading.h"

class Car{
 private:
  /******** Объявление полей класса ********/
  MyString_stat mark;
  MyString_stat manufacturer;
  int yearOfIssue;                  //поле для указания годы выпуска
  int mileage;                      //поле для пробега
  int price;                        //поле для цены
  unsigned int ID;


 public:

  Car();

  static unsigned int descriptor;

  void SetInfo();
  void PrintInfoConsole();
};

void PrintInfoToFile(Car *car, size_t sizeOfarray, std::ofstream &file); .
void GetInfoFromFile(Car &car, std::ifstream &file);                    .
void GetInfoFromFile(Car &car, std::ifstream &file, int numberOfObj);

#endif
```

Файл реализации **car.cpp**

```
#include "car.h"

unsigned int Car::descriptor = 0;

Car::Car(){
    strcpy(mark,"");
    strcpy(manufacturer,"");
    yearOfIssue = 0;
```

```
    mileage = 0;
    price = 0;
    descriptor++;
    ID = descriptor;
}


void Car::SetInfo(){
    char Mark[80];
    char Manufacturer[80];
    char Year[11];
    char Mileage[11];
    char Price[10];


    cout << "------------------- Автомобиль №" << ID << " ----------------------" << endl;

    cout << " Введите марку: ";
    cin.get(Mark,80);
    mark = Mark;
    cin.ignore(80, '\n');


    cout << " Введите изготовителя: ";
    cin.get(Manufacturer,80);
    manufacturer = Manufacturer;
    cin.ignore(80, '\n');


    while(true){
       cout << " Введите год выпуска: ";
       cin >> Year;
            if(cin.good()&&CheckInputNumbers(Year)&&(atoi(Year)>1900 && atoi(Year)<2021)){
                yearOfIssue = atoi(Year);
                cin.ignore(10, '\n');
                break;
            }
        cin.clear();
        cout << red << endl << "\t\tГод введен неверно!" << reset << endl;
        cin.ignore(10, '\n');
    }


    while(true){
       cout << " Введите пробег(только целое кол-во км): ";
       cin >> Mileage;
            if(cin.good() && CheckInputNumbers(Mileage) && atoi(Mileage) >= 0){
                mileage = atoi(Mileage);
                cin.ignore(10, '\n');
                break;
            }
        cin.clear();
        cout << red << endl << "\t\tПробег введен неверно!" << reset << endl;
        cin.ignore(10, '\n');
    }


    while(true){
        cout << " Введите цену (в рублях без копеек): ";
        cin >> Price;
            if(cin.good() && CheckInputNumbers(Price) && atoi(Price) > 0){
```

```
                price = atoi(Price);
                cin.ignore(10, '\n');
                break;
            }
          cin.clear();
          cout << red << endl << "\t\tЦена введена неверное!" << reset << endl;
          cin.ignore(10, '\n');
    }
}


void Car::PrintInfoConsole(){
    cout << "-------------- Автомобиль №" << ID << " -------------------------" << endl;
    cout << "| \tМарка: " << alignLinesOfChar(mark,20);
    cout << "| \tГод выпуска:  " << yearOfIssue << "  ";
    cout << "| \tПробег(км): " << std::setw(8) << mileage <<"  ";
    cout << "| \tИзготовитель: " << alignLinesOfChar(manufacturer,30);
    cout << "| \tЦена: " << std::setw(8) << price;
    cout << "| \tID: " << std::setw(3) << ID << " | " << endl;
}




void PrintInfoToFile(Car *car, size_t sizeOfarray, std::ofstream &file){
     for(int i=0; i < sizeOfarray; i++){
         file.write(reinterpret_cast<char*>(&car), sizeof(*car));
     }
};

void GetInfoFromFile(Car &car, std::ifstream &file){

    file.seekg(0, std::ios::end);
    int endposition = file.tellg();
    int counter = endposition / sizeof(car);
    if(counter == 0){
        cout << red << endl << "\t\t\t\tЧитаемый файл пуст!!!" << reset << endl;
        cout << red << "==   Досрочное завершение программы...   ==" << reset << endl << endl;
        exit(1);
    }

    PrintLogotip();
    PrintHat(3, 12);

    cout << " \tЕсли во время вывода таблица " << magenta << "\"съехала\"" << reset << ", то
                        перезаполните таблицу согласно её размерам." << reset << endl << endl;
    cout << endl << "\t\t\tВсего автомобилей в таблице: " << magenta << counter << reset <<
                                                                                    endl << endl;


    int position;


    for(int i=0; i < counter; i++){
        position = (i)*sizeof(car);
        file.seekg(position);
        file.read(reinterpret_cast<char*>(&car), sizeof(car));
        car.PrintInfoConsole();
    }
    cout << "-------------------------------------------------------------------" << endl;
}
```

```
void GetInfoFromFile(Car &car, std::ifstream &file, int numberOfObj){

    file.seekg(0, std::ios::end);
    int endposition = file.tellg();
    int counter = endposition / sizeof(car);
    if(counter == 0){
        cout << red << endl << "\t\t\t\tЧитаемый файл пуст!!!" << reset << endl;
        cout << red << "==   Досрочное завершение программы...   ==" << reset << endl << endl;
        exit(1);
    }


    if(numberOfObj > counter){
        cout << red << endl << "\t\t\tВ файле нет указанного кол-ва пунктов!" << reset <<
                                                                                    endl;
        cout << "\t\t\t       Кол-во пунктов в файле: " << counter << endl;
        cout << red << "==   Досрочное завершение программы...   ==" << reset << endl << endl;
        exit(2);
    }

    if(numberOfObj < 1){
        cout << red << "\t\t\tВведено неверное кол-во пунктов для чтения!" << reset << endl <<
                                                                                    endl;
        cout << flicker << red << "Досрочное завершение программы..."<< reset << endl << endl;
        exit(2);
    }

    PrintLogotip();
    PrintHat(3, 12);

    cout << "  Если во время вывода таблица " << magenta << "\"съехала\"" << red << ", то
            перезаполните таблицу её размерсноти согласно размерам." << reset << endl << endl;
    cout << endl << "\t\t\tВсего автомобилей в таблице: " << magenta << counter << reset <<
                                                                        endl << endl;


    int position;

    for(int i=0; i < numberOfObj; i++){
        position = (i)*sizeof(car);
        file.seekg(position);
        file.read(reinterpret_cast<char*>(&car), sizeof(car));
        car.PrintInfoConsole();
    }
    cout << "-----------------------------------------------------------------" << endl;
}
```

Библиотека heading

Заголовочный файл **heading.h**

```
#ifndef _HEADER_H_

#define _HEADER_H_

#include <iostream>
#include <cstring>
#include <fstream>
```

```
#include <iomanip>
#include <cstdlib>
#include <string>


using std::cin;
using std::cout;
using std::endl;


const std::string red("\x1B[0;31m");
const std::string yellow("\x1B[1;33m");
const std::string cyan("\x1B[0;36m");
const std::string magenta("\x1B[0;35m");
const std::string reset("\x1B[0m");
const std::string flicker("\x1B[0;5m");


int CheckInputNumbers(char *str);

std::string alignLinesOfChar(std::string str, int width);

void PrintLogotip();
void PrintHat(const int jubNumber, const int optionNumber);
void PrintShortreferens();
void PrintFullReferens(const double release, const double update);
void PrintErrorFlag();
void PrintWarning();


int GetQuantityOfPoints();
int GetQuantityOfPoints(char *argv);


void CheckFile(std::ofstream &file);
void CheckFile(std::ifstream &file);


#endif
```

Файл реализации **heading.cpp**

```
#include "heading.h"


void CheckFile(std::ofstream &file){  //на запись
     if(!file.is_open()){
          cout << red << endl <<  "\t\t\tПроизошла ошибка при открытие файлового потока для
                                                                    записи!" << endl << endl;
          cout << "\t\t\t\tУстраните проблему и перезапустите программу!" << reset << endl;
          cout << flicker << red << "\t\tДосрочное завершение программы... \t" << reset <<
                                                                          endl << endl;
          exit(1);
     }
}

void CheckFile(std::ifstream &file){  //на чтение
     if(!file.is_open()){
          cout << red << endl << "\t\t\tПроизошла ошибка при открытии файлового потока для
                                                                       чтения!" << endl;
          cout << "\t\t\t\tУстраните проблему и перезапустите программу!" << reset << endl;
          cout << flicker << red << "Досрочное завершение программы..." << reset << endl <<
```

```
                                                                                    endl << endl;
          exit(1);
     }
}


int CheckInputNumbers(char *str){
     size_t len = strlen(str);
     for (int i = 0; i < len; i++){
          if(str[i] < '0' || str[i] > '9'){
               return 0;
          }
     }
     return 1;
};



int GetQuantityOfPoints(){
     char numberOfLines[12];
     int NumbLines;

     cout << "\tБыл выбран режим без указания кол-ва пунктов в таблице при запуске
                                                                                  программы.";

     while(true){
          cout << endl << flicker << "\t\tУкажите кол-во пунктов в таблице: " << reset;
          cin.unsetf(std::ios::skipws);
          std::cin >> numberOfLines;
          if(cin.good() && atoi(numberOfLines) > 0 && CheckInputNumbers(numberOfLines)){
               if(atoi(numberOfLines) > 100){
                    cout << endl << red << "\t\t\t\t\tПревышен лимит записей!" << reset <<
                                                                                       endl;
                    cout << "\t\t\t\tВряд ли вам надо так много пунктов :) " << endl;
                    cout << "\t\tСкорее всего такое кол-во пунктов(или более) приведет к
                                                  ошибке или зависанию системы." << endl << endl;
                    cout << flicker << red << "================\tДосрочное завершение
                                   программы... \t================" << reset << endl << endl;
                    exit(1);
               }
               NumbLines = atoi(numberOfLines);
               cin.ignore(10, '\n');
               break;
          }
          cin.clear();
          cout << red << endl << "\t\tНеправильный ввод! Введите корректное число пунктов
                                                                              таблицы." << reset;
          cin.ignore(10, '\n');
     }
     cout << std::endl;
     return NumbLines;
}


int GetQuantityOfPoints(char *argv){
     if(atoi(argv) < 1 || !CheckInputNumbers(argv)){
          cout << red << endl << "\t\tБыло задано неверное кол-во пунктов при запуске!" <<
                                                                                       endl;
          cout << flicker << red << "Досрочное завершение программы..." << reset << endl <<
                                                                                       endl;
          exit(1);
```

```
    }
    int numberOfLines = std::atoi(argv);
    return numberOfLines;
};

std::string alignLinesOfChar(std::string str, int width){

    int len = str.length();
    if(len > width){return str;};

    int diff = width -len;
    int pad1 = diff/2;
    int pad2 = diff - pad1;

    return std::string(pad1,' ') + str + std::string(pad2,' ');
};

void PrintErrorFlag(){
    cout << red << endl << "\t\t\tБыл введен неверный флаг при запуске!" << endl;

    cout << " Для получении справки о флагах запуска воспользуйтесь справкой: ./lab3 [-h || -
                                                                                -help]" << endl;
    cout  << "== Досрочное завершение программы...=="<< reset << endl << endl;
};

void PrintWarning(){
    cout << red << "  Внимание!!! " << reset << "При вводе Марки/Изготовителя на русском
                        языке, во время вывода таблицы в консоль она может " << magenta <<
                                                            "\"съехать\"" << reset << endl;
    cout << "\t\t\t\tРекомедуемый язык для ввода: " << cyan << "Английский" << reset << endl
                                                                                << endl;
};
```

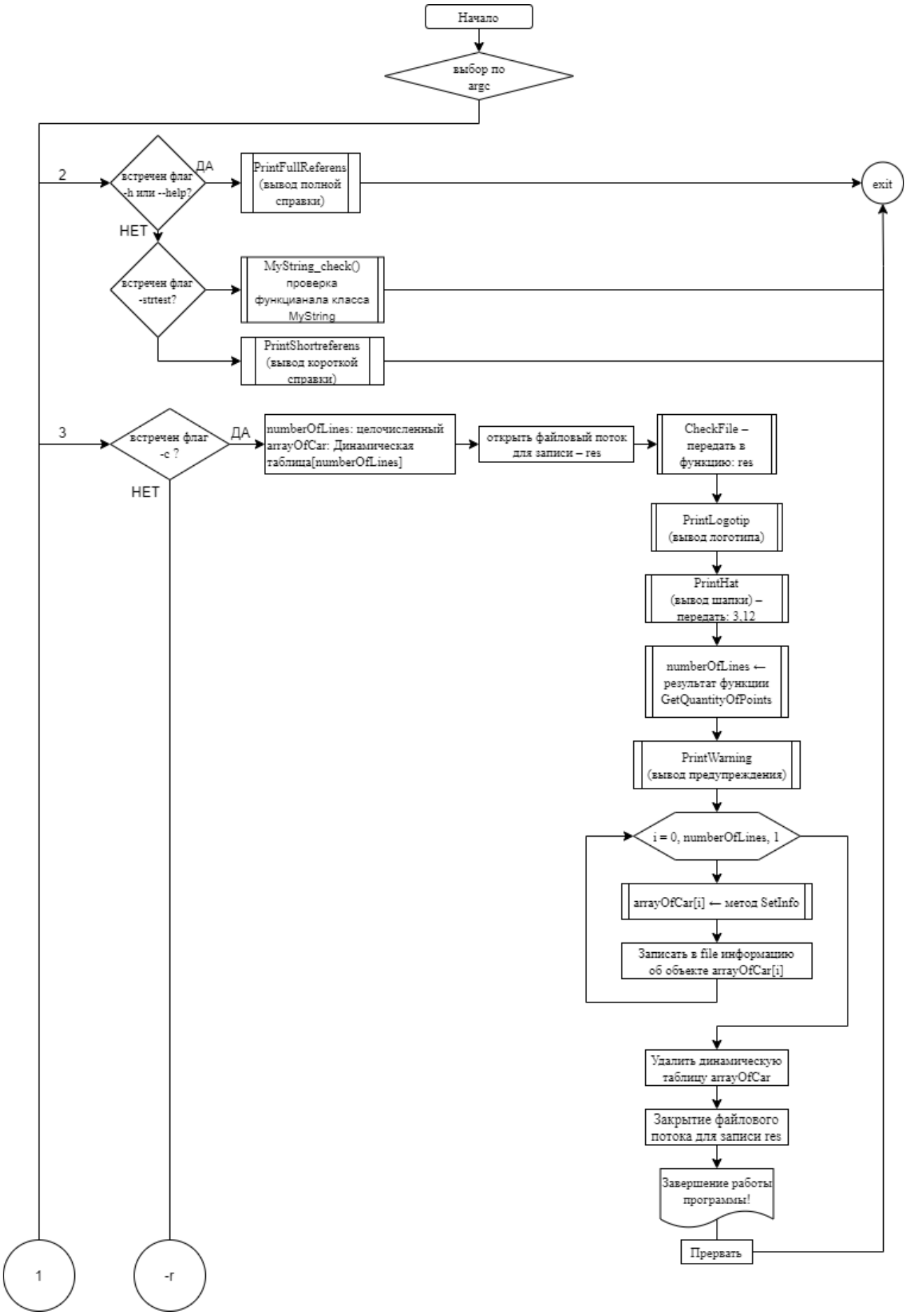
Функция main () – вход в программу

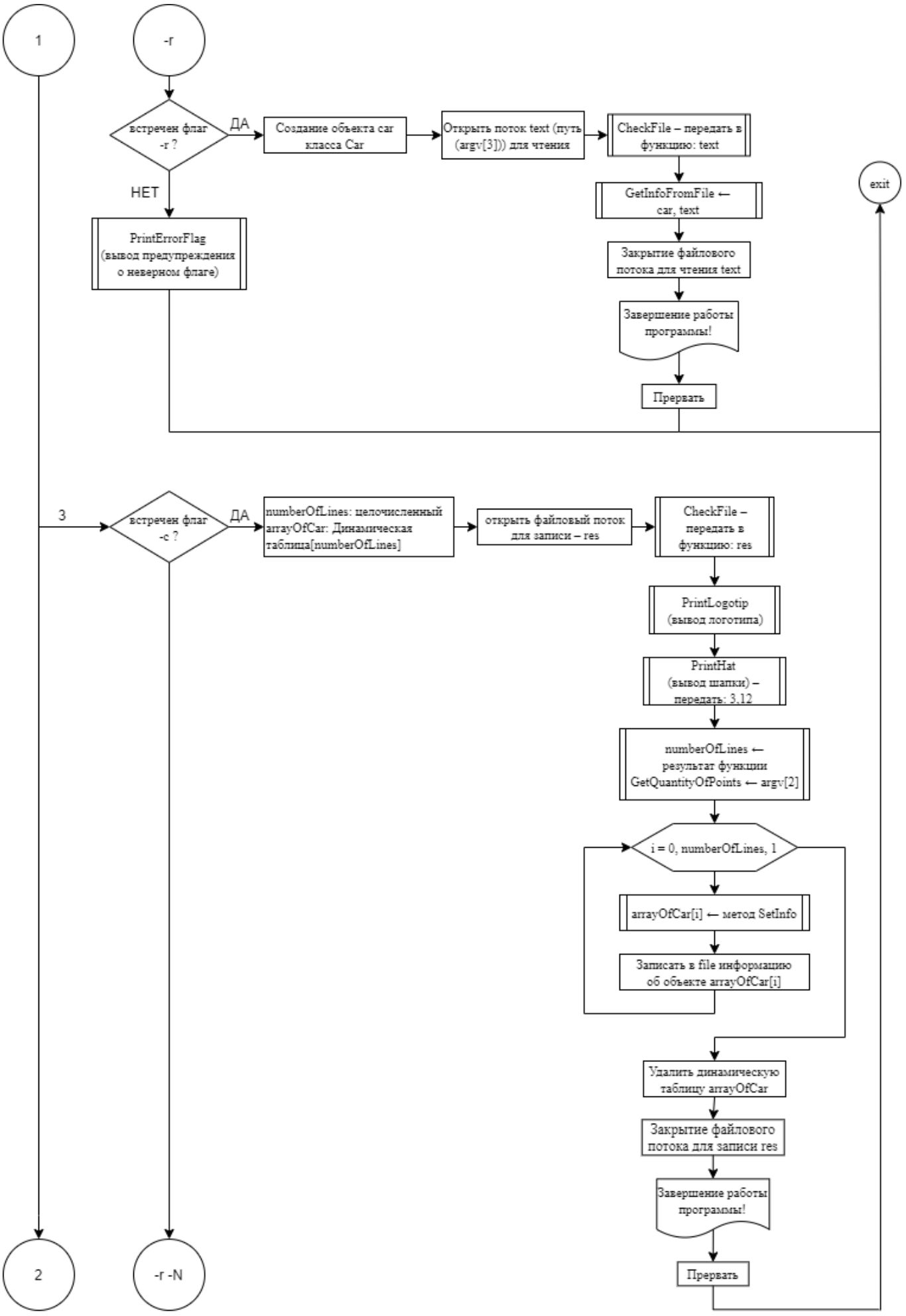
```
#include "car.h"

int main (int argc, char* argv[]){

  switch (argc){
    case 2:{
      if(!strcmp(argv[1],"-h") || !strcmp(argv[1],"--help"))
      {
        PrintFullReferens(08.04, 12.04);
        exit(0);
      };
      PrintErrorFlag();
      break;
    }

    case 3:{
      if (!strcmp(argv[1],"-c")){

        std::ofstream res(argv[2], std::ofstream::binary);
        CheckFile(res);

        PrintLogotip();
        PrintHat(3, 12);

        int numberOfLines = GetQuantityOfPoints();
```

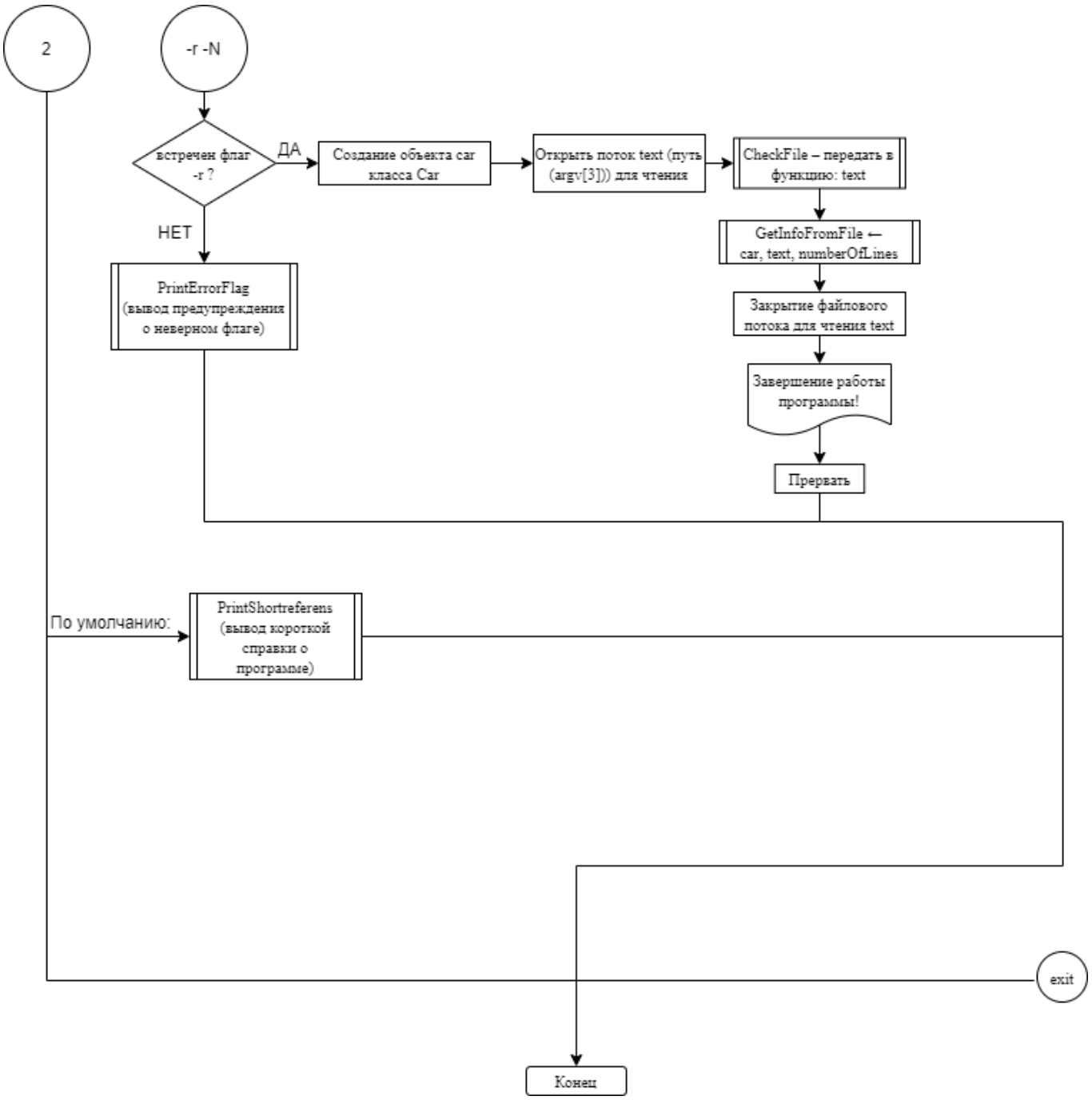
```
    Car *arrayOfCar = new Car[numberOfLines];
    PrintWarning();
    for(int i = 0; i < numberOfLines; i++){
      arrayOfCar[i].SetInfo();
      res.write(reinterpret_cast<char*>(&arrayOfCar[i]), sizeof(arrayOfCar[i]));
    }

    delete[] arrayOfCar;
    res.close();
    cout << endl << red << "\t\tЗавершение работы программы!" << reset << endl;
    break;
  }

  if (!strcmp(argv[1],"-r"))
  {
    std::ifstream text(argv[2], std::ifstream::binary);
    CheckFile(text);

    Car car;

    GetInfoFromFile(car, text);

    text.close();
    cout << endl << red << "\t\tЗавершение работы программы!" << reset << endl;
    break;
  }
  PrintErrorFlag();
  break;
}

case 4:{
  if (!strcmp(argv[1],"-c"))
  {
    int numberOfLines = GetQuantityOfPoints(argv[2]);

    std::ofstream res(argv[3]);
    CheckFile(res);

    PrintLogotip();
    PrintHat(3, 12);

    Car *arrayOfCar = new Car[numberOfLines];
    PrintWarning();
    for(int i = 0; i < numberOfLines; i++){
      arrayOfCar[i].SetInfo();
      res.write(reinterpret_cast<char*>(&arrayOfCar[i]), sizeof(arrayOfCar[i]));
    }
    res.close();
    cout << endl << red << "\t\tЗавершение работы программы!" << reset << endl;
    break;
  }

  if (!strcmp(argv[1],"-r"))
  {
    std::ifstream text(argv[3]);
    CheckFile(text);

    int numberOfLines = GetQuantityOfPoints(argv[2]);

    Car car;
```

```
        GetInfoFromFile(car, text, numberOfLines);

        text.close();//закрытие файла
        cout << endl << red << "\t\tЗавершение работы программы!" << reset << endl;
        break;
      }
      PrintErrorFlag();
      break;
    }

    default:{
      PrintShortreferens();
      exit(1);
      break;
    }
  }
  return 0;
}
```

Блок-Схема:


Начало
выбор по argc
2
встречен флаг -h или --help?
ДА
PrintFullReferens (вывод полной справки)
exit
НЕТ
встречен флаг -strtest?
MyString_check() проверка функцианала класса MyString
PrintShortreferens (вывод короткой справки)
3
встречен флаг -c ?
ДА
numberOfLines: целочисленный arrayOfCar: Динамическая таблица[numberOfLines]
открыть файловый поток для записи – res
CheckFile – передать в функцию: res
PrintLogotip (вывод логотипа)
PrintHat (вывод шапки) – передать: 3.12
numberOfLines ← результат функции GetQuantityOfPoints
PrintWarning (вывод предупреждения)
i = 0, numberOfLines, 1
arrayOfCar[i] ← метод SetInfo
Записать в file информацию об объекте arrayOfCar[i]
Удалить динамическую таблицу arrayOfCar
Закрытие файлового потока для записи res
Завершение работы программы!
Прервать
НЕТ
1
-r


1
-r
встречен флаг -r ?
ДА
НЕТ
Создание объекта car класса Car
Открыть поток text (путь (argv[3])) для чтения
CheckFile – передать в функцию: text
GetInfoFromFile ← car, text
Закрытие файлового потока для чтения text
Завершение работы программы!
Прервать
PrintErrorFlag (вывод предупреждения о неверном флаге)
exit
3
встречен флаг -c ?
ДА
numberOfLines: целочисленный arrayOfCar: Динамическая таблица[numberOfLines]
открыть файловый поток для записи – res
CheckFile – передать в функцию: res
PrintLogotip (вывод логотипа)
PrintHat (вывод шапки) – передать: 3,12
numberOfLines ← результат функции GetQuantityOfPoints ← argv[2]
i = 0, numberOfLines, 1
arrayOfCar[i] ← метод SetInfo
Записать в file информацию об объекте arrayOfCar[i]
Удалить динамическую таблицу arrayOfCar
Закрытие файлового потока для записи res
Завершение работы программы!
Прервать
2
-r -N



Скриншоты:

Неверный флаг при запуске:

```
[vladislav@localhost Лабораторная №3]$ ./main -t text

                    Был введен неверный флаг при запуске!
 Для получении справки о флагах запуска воспользуйтесь справкой: ./lab3 [-h || --help]
======================     Досрочное завершение программы...     ======================
```

Флаг –strtest при запуске:

```
        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: house
1ая строка:
  home

2ая строка:
  house

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 1
 строка №1 не равна строке №2
[vladislav@localhost Лабораторная №4]$ ./main -strtest

        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: home
1ая строка:
  home

2ая строка:
  home

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 1
 строка №1 равна строке №2
```

```
        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: house
1ая строка:
  home

2ая строка:
  house

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 2
 строка №1 не равна строке №2
[vladislav@localhost Лабораторная №4]$ ./main -strtest

        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: home
1ая строка:
  home

2ая строка:
  home

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 2
 строка №1 не равна строке №2
```

```
        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: house
1ая строка:
  home

2ая строка:
  house

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 3
 строка №1 не больше строки №2
[vladislav@localhost Лабораторная №4]$ ./main -strtest

        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: house
Введите содержимое 2ой строки: home
1ая строка:
  house

2ая строка:
  home

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 3
 строка №1 больше строки №2
```

```
        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: house
1ая строка:
  home

2ая строка:
  house

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 4
 строка №1 меньше строки №2
[vladislav@localhost Лабораторная №4]$ ./main -strtest

        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: house
Введите содержимое 2ой строки: home
1ая строка:
  house

2ая строка:
  home

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 4
 строка №1 не меньше строки №2
```

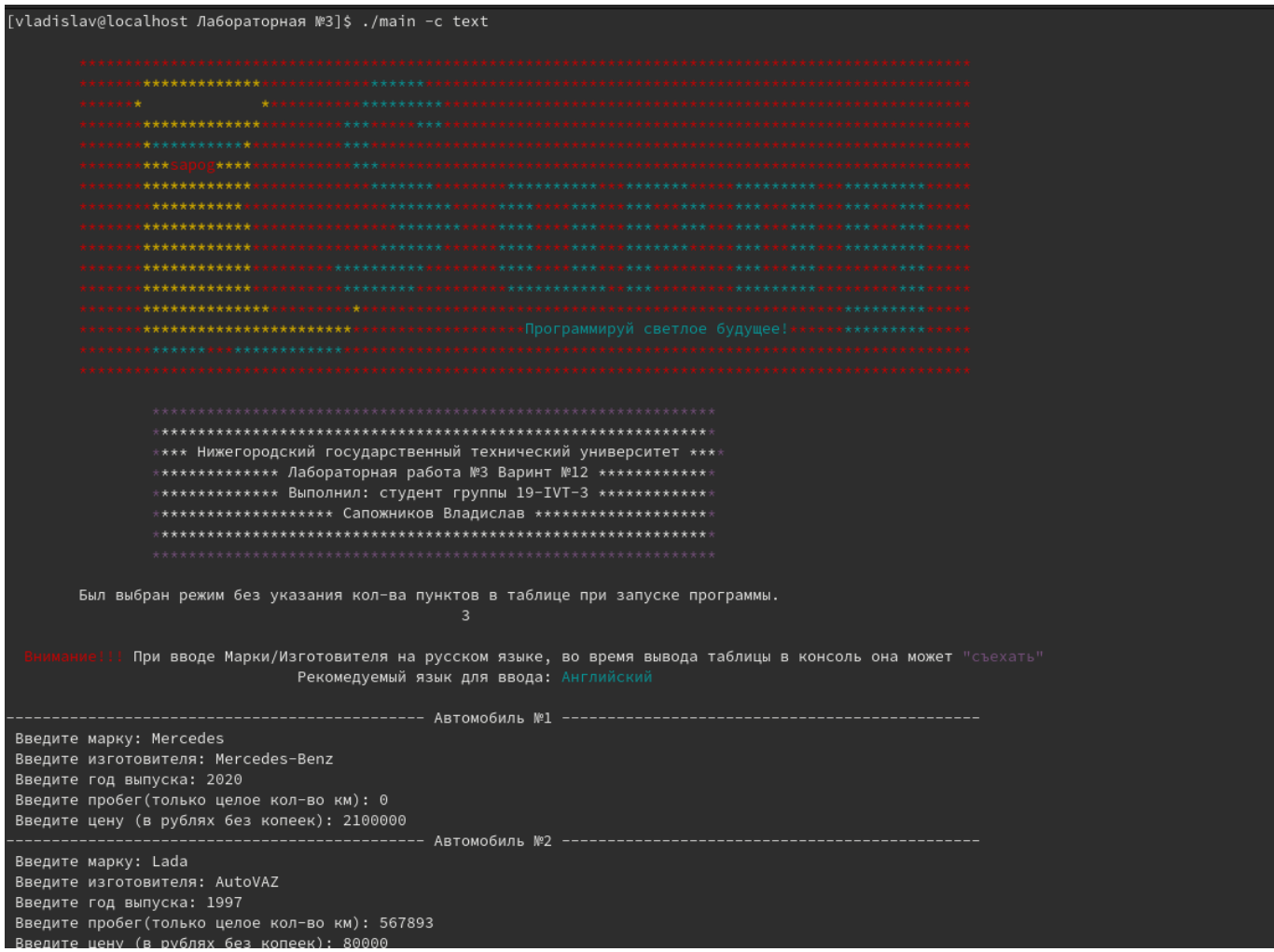
```
[vladislav@localhost Лабораторная №4]$ ./main -strtest

        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: house
1ая строка:
  home

2ая строка:
  house

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 5
 Результирующая строка:
home house
```

```
[vladislav@localhost Лабораторная №4]$ ./main -strtest

        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: house
1ая строка:
  home

2ая строка:
  house

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 6
 Выберите номер строки с которой хотите работать: 1
 Введите номер элемента к которому хотите получить доступ(начало отсчета индексов с 1):
3
Выбранный элемент: m
```

```
[vladislav@localhost Лабораторная №4]$ ./main -strtest

        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: house
1ая строка:
  home

2ая строка:
  house

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 6
 Выберите номер строки с которой хотите работать: 2
 Введите номер элемента к которому хотите получить доступ(начало отсчета индексов с 1):
100
Выбранный элемент:
Ошибочный индекс: 100! Выход за пределы массива!!!
Исправте ошибку и попробуйте еще раз!
```

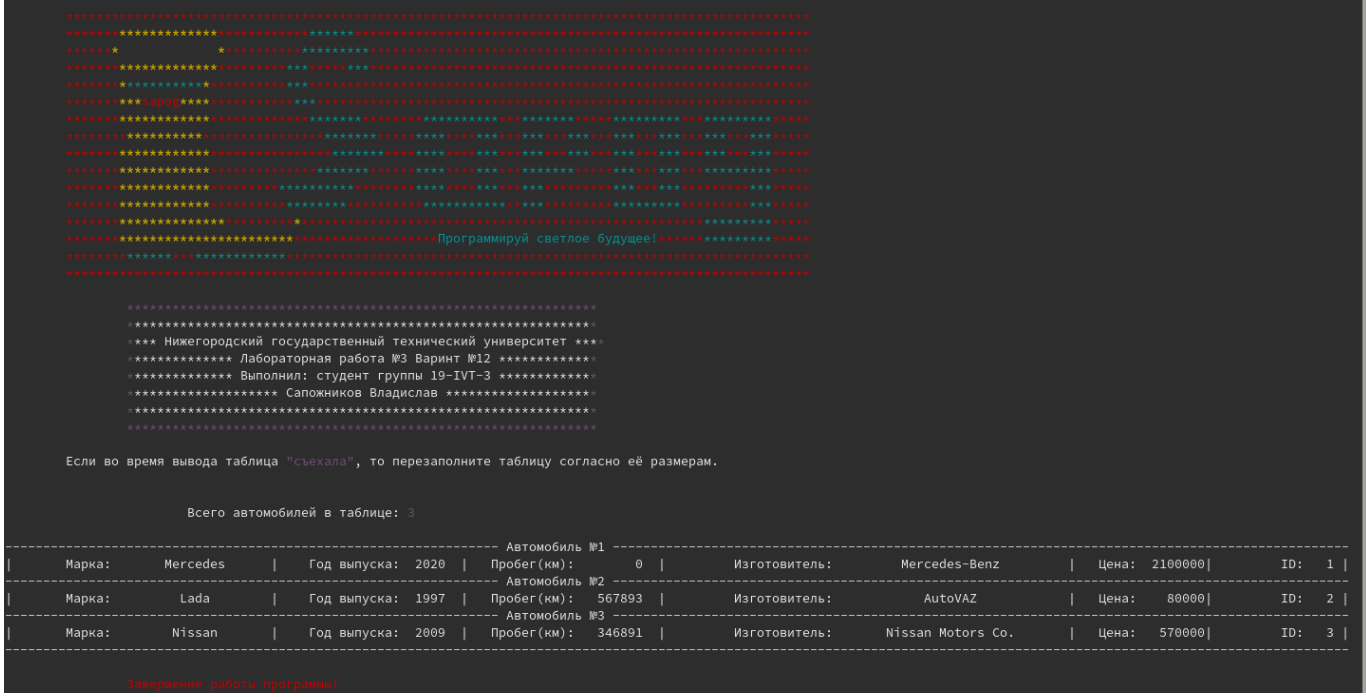
```
[vladislav@localhost Лабораторная №4]$ ./main -strtest

        Класс MyString - динамическая строка char

Warning!!! рекомендуемый язык для ввода: English

Введите содержимое 1ой строки: home
Введите содержимое 2ой строки: house
1ая строка:
  home

2ая строка:
  house

Выберите оператор для проверки:
        1 - "=="
        2 - "!="
        3 - ">"
        4 - "<"
        5 - "+"
        6 - "at()"
        7 - "[]"
Ваш выбор: 7
 Выберите номер строки с которой хотите работать: 2
 Введите номер элемента к которому хотите получить доступ(начало отсчета индексов с 1):
3
Выбранный элемент: u
```

Флаг –c при запуске:



Флаг –r при запуске:

```
[vladislav@localhost Лабораторная №3]$ ./main -r text
```




```
      Если во время вывода таблица "съехала", то перезаполните таблицу согласно её размерам.


                    Всего автомобилей в таблице: 3

---------------------------------------------------------- Автомобиль №1 ----------------------------------------------------------
|       Марка:       Mercedes      |    Год выпуска:  2020  |   Пробег(км):        0  |         Изготовитель:         Mercedes-Benz         |   Цена:  2100000|         ID:   1 |
---------------------------------------------------------- Автомобиль №2 ----------------------------------------------------------
|       Марка:         Lada        |    Год выпуска:  1997  |   Пробег(км):   567893  |         Изготовитель:           AutoVAZ           |   Цена:    80000|         ID:   2 |
---------------------------------------------------------- Автомобиль №3 ----------------------------------------------------------
|       Марка:        Nissan       |    Год выпуска:  2009  |   Пробег(км):   346891  |         Изготовитель:       Nissan Motors Co.       |   Цена:   570000|         ID:   3 |
-----------------------------------------------------------------------------------------------------------------------------------


              Завершение работы программы!
```

Флаг –c с указанным кол-ом пунктов при запуске:

```
[vladislav@localhost Лабораторная №3]$ ./main -c 2 text
```


```
  Внимание!!! При вводе Марки/Изготовителя на русском языке, во время вывода таблицы в консоль она может "съехать"
                              Рекомедуемый язык для ввода: Английский

------------------------------------------------ Автомобиль №1 ------------------------------------------------
 Введите марку: Mercedes
 Введите изготовителя: Mercedes-Benz
 Введите год выпуска: 2020
 Введите пробег(только целое кол-во км): 0
 Введите цену (в рублях без копеек): 2100000
------------------------------------------------ Автомобиль №2 ------------------------------------------------
 Введите марку: Lada
 Введите изготовителя: AutoVAZ
 Введите год выпуска: 1997
 Введите пробег(только целое кол-во км): 456789
 Введите цену (в рублях без копеек): 80000

                Завершение работы программы!
```

Флаг –r с указанным кол-ом пунктов при запуске: