Федеральное государственное автономное образовательное учреждение   
высшего образования «Национальный исследовательский технологический   
университет «МИСиС»

Институт информационных технологий и компьютерных наук

Кафедра Инженерной Кибернетики

КУРСОВАЯ РАБОТА   
по дисциплине «Технологии программирования»  
на тему:  
«Консольный архиватор»

Разработал:

студент группы БПМ-19-2

Котелевский Артём Алексеевич

Проверил:

доцент кафедры ИК, к.т.н

Полевой Д.В.

Москва, 2020

**СОДЕРЖАНИЕ**

1 Описание задачи 3

1.1 Общие положения 3

1.2 Назначение разработки 3

1.3 Требования к программе 3

2 Пользовательское описание 4

2.1 Общие положения 4

2.2 Пользовательский интерфейс 4

3 Техническое описание 6

3.1 Классы и функции 6

3.1.1 Класс archivator\_info 6

3.1.2 Класс command\_line 7

3.1.3 Класс file\_header 8

4 Cборка 9

5. Тесты 10

5.1 Проверка работоспособности 10

5.2 Проверка создания архива 10

5.3 Проверка размера файла, заархивированного в 5.2 10

5.4 Проверка правильности разархивирования.................................................................10

ПРИЛОЖЕНИЕ А Проверка работоспособности на реальном примере 11

1. Описание задачи
   1. Общие положения

Консольное приложение «Архиватор»(Далее - КП) включает в себя возможность использовать различные алгоритмы шифрования данных(в данной работе XOR шифрование).

КП предназначено для обьединения файлов в архив, добавление новый файлов по мере их поступления, удаления файлов и разархивации всех файлов.

* 1. Назначение разработки

КП реализует набор классов, обеспечивающих следующие функции:

* Считывание данных из файла;
* Шифрование содержимого файла;
* Добавление и удаление файлов из архива;
* Получение списка файлов в архиве
* Разархивация файлов.
  1. Требования к программе

Библиотека должна быть разработана в виде библиотеки классов на языке C++. Стандарт языка C++ не менее ISO C++17.

Средство сборки библиотеки: CMake (версии не менее 3.18), среда разработки: Microsoft Visual Studio 2019.

1. Пользовательское описание
   1. Общие положения

КП реализует функционал архиватора для работы с файлами. Программа имеет 4 команды:

1) “-a” - добавление файлов в уже существующий архив или создание такого при его отсутствии.

2) “-d” – удаление файлов из уже существующего архива

3)”-l” – вывод списка файлов в архиве

4) “-x” – разархивация файлов из архива

Пример работы программы:

-a file.arch testdata.sql script+insert.sql tmp.txt

-d file.arch testdata.sql

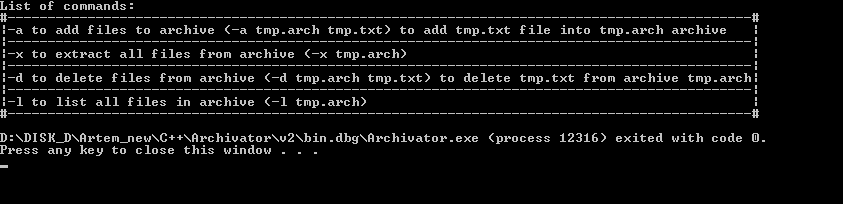
-l file.arch

-x file.arch

Команды передаются в КП посредством Command arguments в настройках Debugger-а, либо при переходе в директорию с исполняемым файлом. Первым параметром всегда вводится команда, далее идёт имя архива, после этого в зависимости от команды могут быть переданы имена файлов.

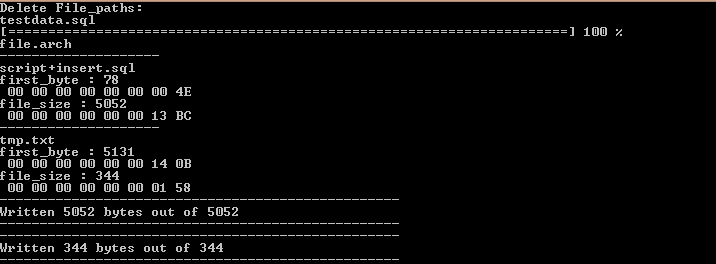
* 1. Пользовательский интерфейс

Если пользователь не знает как работают комманды, либо введена неправильная команда будет выведен список команд с примерами:

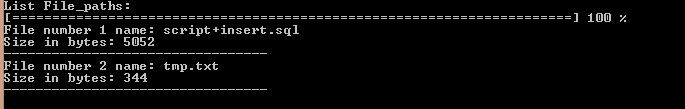


При добавлении файлов(“-a”) в архив пользователь увидит информацию о файлах в архиве (их имена, размер и первый байт записи файла в архиве) 

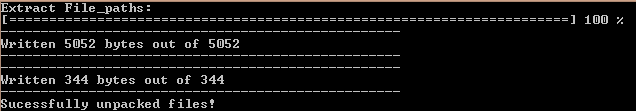
При удалении файла из архива(“-d”) пользователь получит имена, размер и первый байт записи файла в архиве.



При вызове команды “-l” пользователь получит список файлов в архиве, а также их размер



При вызове команды “-x” пользователь получит разархивированные файлы в папке с проектом, а также сообщение о статусе разархивации.



1. Техническое описание
   1. Классы и функции

Обьявление классов находится в файле «Archiver.h». Структура файла приведена в таблице 1.

Таблица 1 – Структура файла «Archiver.h»

|  |  |
| --- | --- |
| Наименование | Описание |
| command\_line | Класс работы с командной строкой |
| file\_header | Класс работы с файлом для получения информации о нём |
| archivator\_info | Класс работы с архивированным файлом |

3.1.1 Класс archivator\_info

### Открытые члены

* void **get\_file\_info** (const std::string &file\_name, **file\_header** &fh)

*получение информации о файле, размер, имя*

* std::string **get\_file\_name** (std::string const &path)

*метод получения имени из пути файла*

* std::ifstream::pos\_type **get\_file\_size** (const std::string &filename)

*получает размер файла*

* void **construct\_header** (std::vector< std::string > &file\_paths, std::vector< **file\_header** > &headers, bool flag)

*принимает команды*

* void **pack** (const **command\_line** &cl)

*архивация файлов*

* void **unpack** (const **command\_line** &cl)

*разархивация файлов*

* void **delete\_file** (const **command\_line** &cl)

*удаление файлов*

* void **list** (const **command\_line** &cl)

*список файлов в архиве*

* void **write\_header** (std::ofstream &out, const std::vector< **file\_header** > &headers)

*метод архивации файла*

* void **write\_file** (std::ofstream &out, std::string file\_name)

*метод записи из файла в архив*

* void **get\_header\_info** (std::ifstream &in, std::vector< **file\_header** > &files)

*получение информации из заголовка*

* void **rewrite\_files** (std::ifstream &in, std::ofstream &out, **file\_header** file, bool action)
* void **int64ToChar** (unsigned char mesg[], int64\_t num)

*перевод из int64 к символьному типу*

* void **XOR\_cipher\_decipher** (char &orignal\_char)

*метод XOR шифрования*

* int64\_t **charTo64bitNum** (unsigned char a[])

*перевод из символьного типа к int64*

### Открытые статические члены

* static void **check\_file** (std::string file\_name)

*метод проверки существования файла*

Закрытые данные

* std::vector<file\_header> file\_info;

*вектор, хранящий информацию о файлах*

* int file\_amount = 0;

*кол-во файлов*

* char signature[6] = "art25";

*сигнатура архива*

Объявления и описания членов классов находятся в файлах:

* Archiver.h
* Archiver.cpp

3.1.2 Класс command\_line

Открытые члены

* **command\_line** ()

*конструктор*

* command\_t **get\_command** ()

*получает нужную комманду*

* const std::string & **get\_archive\_name** () const

*получает имя архива*

* const std::vector< std::string > & **get\_file\_names** () const

*получает имена файлов*

* void **set\_file\_names** (std::vector< std::string > file\_names)

*заполняет вектор именами файлов, для работы с ними*

* void **set\_archive\_name** (std::string archive\_name)

*устанавливает имя архива*

* bool **parse** (int argc, char \*argv[])
* void **parse\_file\_names** (int argc, char \*argv[])

*считывает имена файлов*

* void **print** ()
* void **print\_command\_list** ()

*выводит список команд*

### Открытые статические члены

* static void **progress\_bar** ()

*выводит полоску загрузки*

Закрытые члены

* command\_t m\_command;

*какая команда выбрана*

* std::string m\_archive\_name;

*имя архива*

* std::vector<std::string> m\_file\_names;

*имена файлов*

Объявления и описания членов классов находятся в файлах:

* Archiver.h
* Archiver.cpp

3.1.3 Класс file\_header

### Открытые члены

* void **set\_name** (const std::string &name)

*устанавливает имя файла*

* const std::string & **get\_name** ()

*получает имя файла*

* void **set\_size** (int64\_t size)

*устанавливает размер файла*

* int64\_t **get\_size** ()

*получает размер файла*

* void **set\_first\_byte** (int64\_t first\_byte)

*записывает в хедер первый байт файла в архиве*

* int64\_t **get\_first\_byte** ()

*получает из хедера первый байт файла*

Закрытые данные

* std::string m\_name;

*имя файла*

* int64\_t m\_size;

*размер файла*

* int64\_t m\_first\_byte;

*первый байт файла*

Объявления и описания членов класса находятся в файле:

* Archiver.h

4 Сборка

1. Загрузить проект, размещенный в svn-хранилище по ссылке: <https://mysvn.ru/Artem_Kotelevsky/CourseWork/>

2. В программе cmake-gui.exe указать путь до корневой папки проекта, также нужно указать компилятор.

3. После этого введите директорию, в которой вы собираетесь построить проект в поле CMAKE\_INSTALL\_PREFIX.

4. Теперь нужно построить проект.

5. Запускаем через cmake проект и выбираем Archivator как StartUp Project

6. Передаем аргументы через Debug -> Archivator Properties -> Debugging -> Command Arguments

5 Тесты.

5.1 Проверка работоспособности программы.

1. Запустить программу с пустым списком аргументов.

2. Получить список комманд.

3. Закрыть диалоговое окно.

5.2 Проверка создания архива.

1. Запустить программу с командов –a <название вашего архива> <название файла, который вы добавите в папку исполняемого проекта>

2. Закрыть диалоговое окно

3. Найти в папке с исполняемым проектом файл с названием вашего архива

5.3 Проверка размера файла, заархивированного в 5.2.

1. Запустить программу с командой –l <название вашего архива>

2. В диалоговом окне убедиться в правильности размера сохраненного вами файла

5.4 Проверка правильности разархивирования.

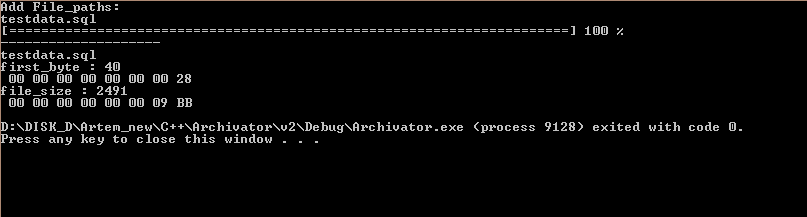
1. Запустить программу с командов –x <название вашего архива>

2. Проверить файл на правильность записи

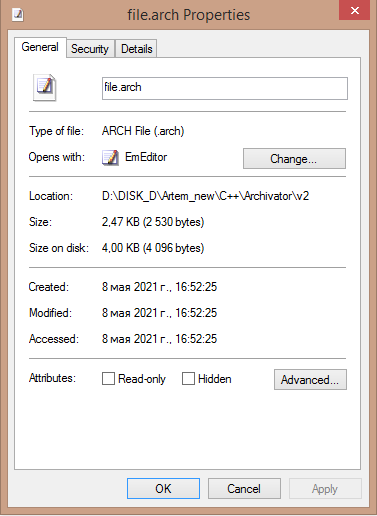
ПРИЛОЖЕНИЕ А

Проверка работоспособности на реальном примере

1) Вводим команду -a file.arch testdata.sql где file.arch имя архива, а testdata.sql имя добавляемого в архив файла



2) Видим, что размер нашего архива 2530. Запись файла начиналась с 40 байта и размер файла 2491 => архив создан верно



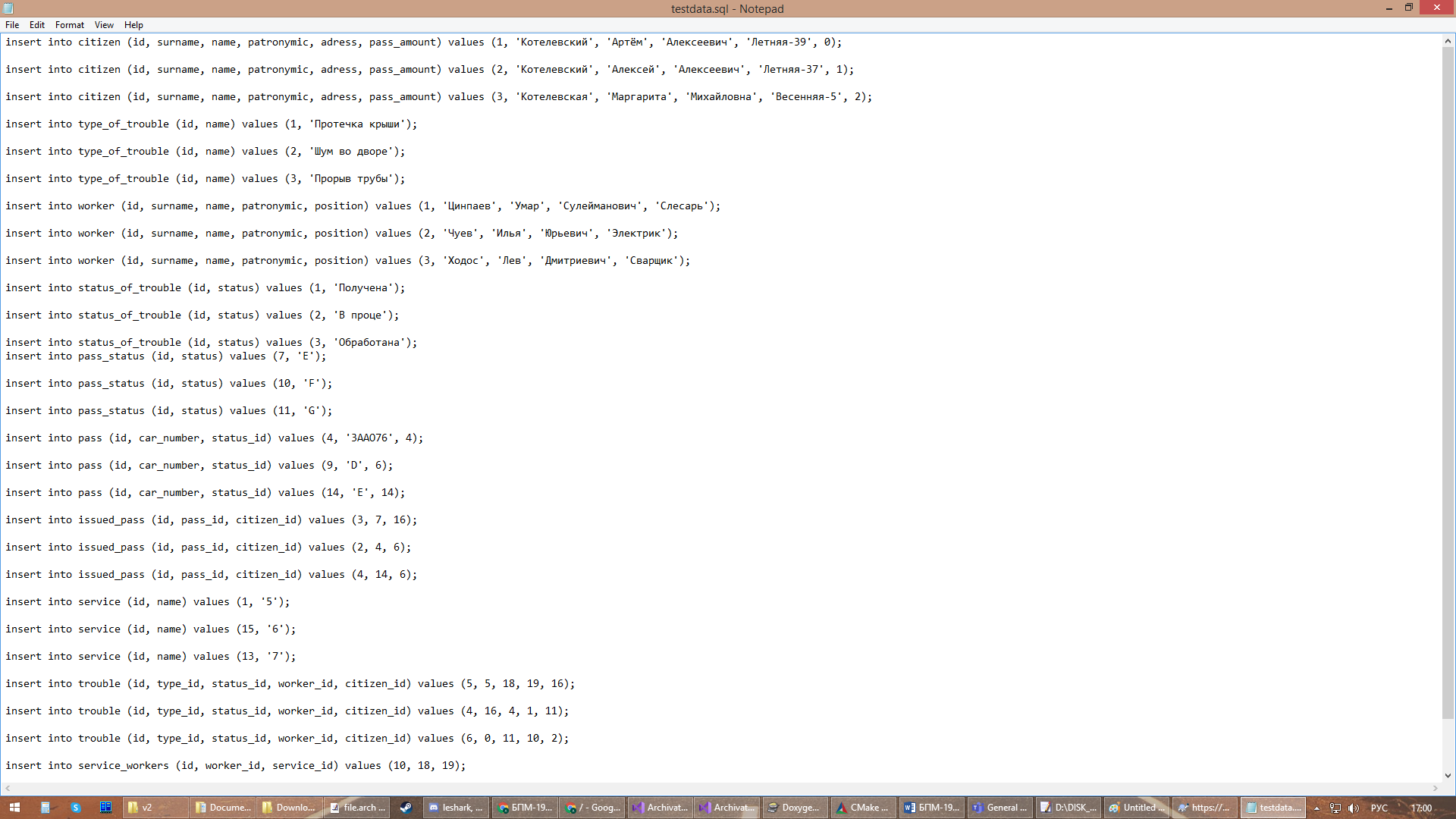
3) Введем команду –l file.arch чтобы проверить правильность записи



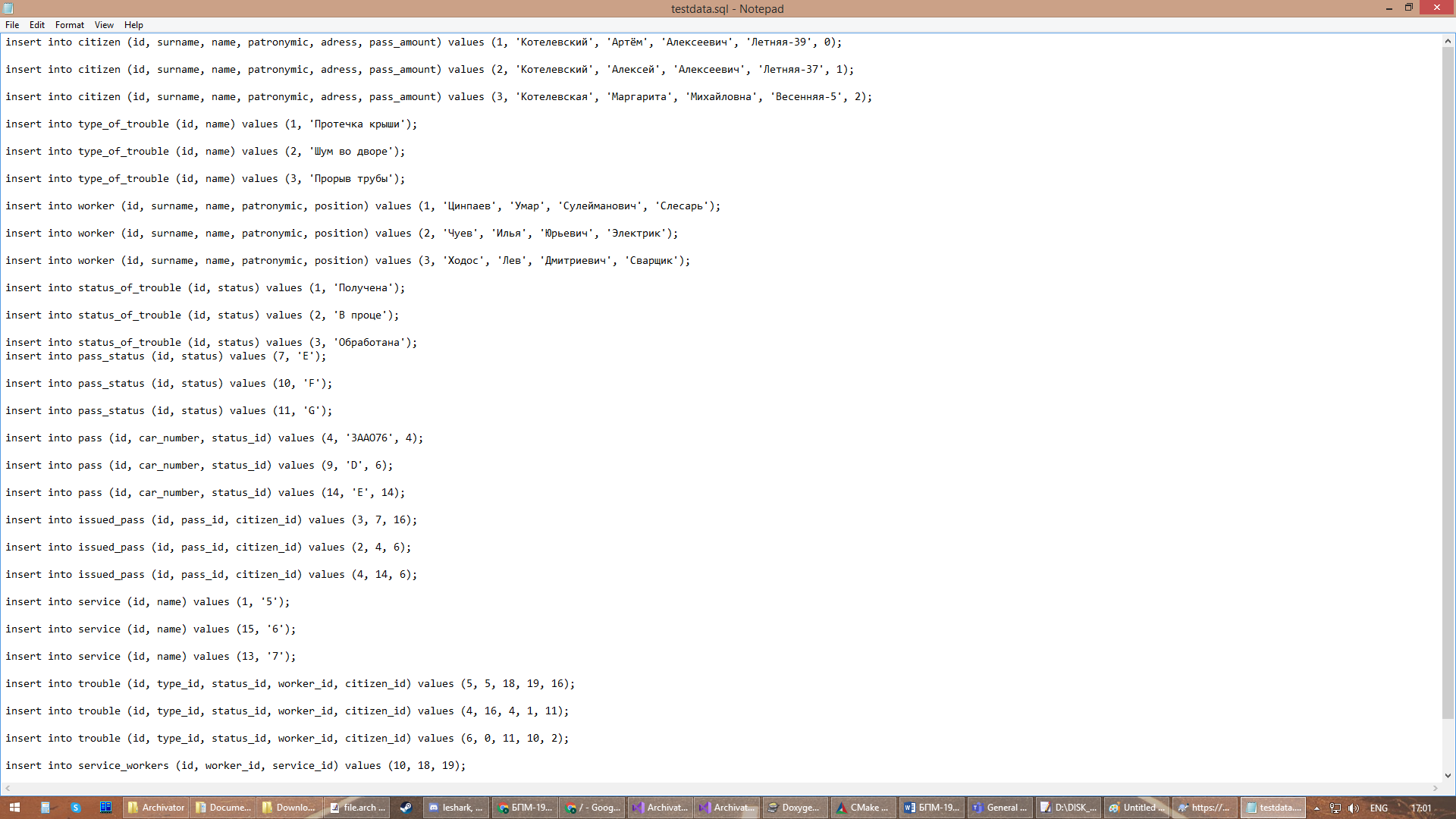
4) Введем команду –x file.arch чтобы разархивировать созданный архив и проверим правильность разархивации



Файл до:



Файл после:



Файл разархивирован правильно, что означает правильность работы программы.