



**Министерство науки и высшего образования Российской Федерации
Федеральное государственное бюджетное образовательное
учреждение
высшего образования
«Московский государственный технический университет
имени Н. Э. Баумана
(национальный исследовательский университет)»
(МГТУ им. Н. Э. Баумана)**

ФАКУЛЬТЕТ
КАФЕДРА

«Информатика и системы управления» (ИУ)
«Информационная безопасность» (ИУ8)

**Домашняя работа № 3
ПО КУРСУ
«Алгоритмические языки»**

Студент

ИУ8-13

(Группа)

С. Е. Матушин,
В. С. Ажгирей,
Е. М. Грязнов

(И. О. Фамилия)

Преподаватель:

М. В. Малахов

(И.О. Фамилия)

Введение

Цели и задачи работы

Цель работы состоит в овладении навыками разработки архитектуры ПО и сборки ПО. Для достижения цели необходимо выполнить следующие задачи:

1. Реализация сборки ПО с использованием `stake` через командную строку.
2. Разработка архитектуры целевого ПО.

Реализация

Входные данные располагаются в файле, значения каждого байта меньше `0x80`. Файл архивируется (сжимается) с помощью разработанной программы: если встречаются несколько подряд повторяющихся байт, то они заменяются на 2 байта, первый из которых байт `0x80` + количества повторений данного байта, а второй - сам повторяющийся байт (например, если идут четыре подряд одинаковых байта `0x70`, то есть `"0x70 0x70 0x70 0x70"`, то они заменяются на `"0x84 0x70"`). На выходе будет получаться сжатый файл, изменённый таким образом. Так же, если надо разархивировать (разжать) файл, то выполняются обратные операции (например, набор байт `"0x84 0x70"` заменяется на `"0x70 0x70 0x70 0x70"`). Соответственно, получится разархивированный файл.

Условия для 1 варианта

Разработайте приложение-архиватор, предназначенное для сжатия файлов, в которых значение каждого байта меньше `0x80`. Сжатие производить следующим методом: если подряд встречается несколько одинаковых байт (например `0x70 0x70 0x70`), то удалять все дубли, а вместо них добавлять их количество + `0x80` (то есть для приведенного случая должно получиться `0x83 0x70`). Приложение должно уметь как архивировать, так и разархивировать файлы указанным методом.

Основная часть

В ходе выполнения задания программа была разбита на несколько файлов, с помощью CMake была проведена сборка этих файлов в файл newapp.exe

Снимки выполнения работы

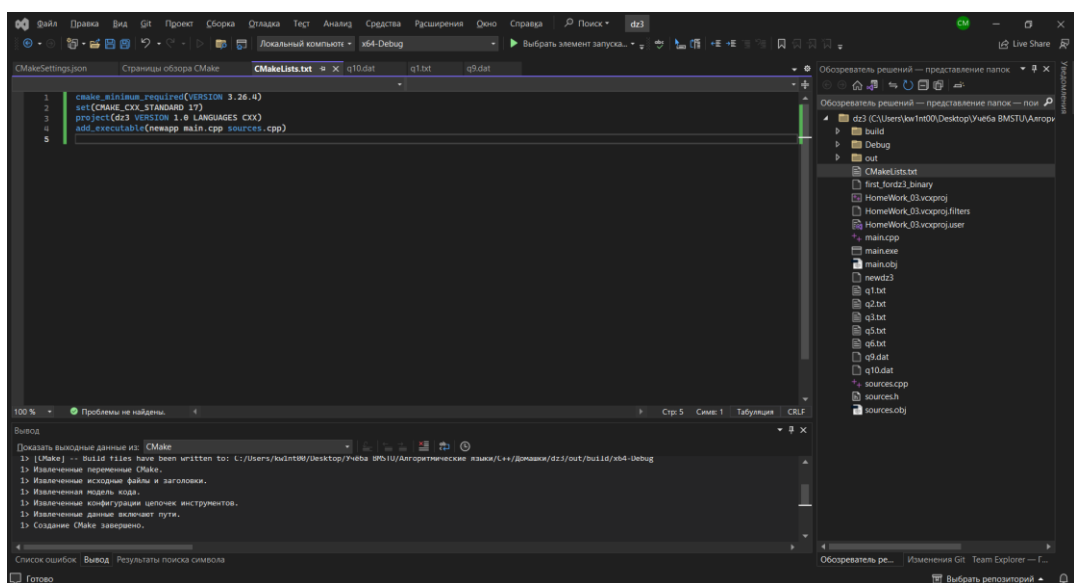


Рисунок 1 – файл CMakeLists.txt

```
MINGW64/c/Users/kwInt00/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/build
C:\Int000DESKTOP-7UTAD4U\MINGW64 ~/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3
$ mkdir build

C:\Int000DESKTOP-7UTAD4U\MINGW64 ~/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3
$ cd
  .\
  CMakeLists.txt      out/
  Debug/              q1.txt
  Homework_03.vcxproj q10.dat
  Homework_03.vcxproj.filters q2.txt
  Homework_03.vcxproj.user q3.txt
  build/              q5.txt
  FirstFordz3.binary  q6.txt
  main.cpp             q9.dat
  main.exe             sources.cpp
  main.obj             sources.h
  newdz3              sources.obj

C:\Int000DESKTOP-7UTAD4U\MINGW64 ~/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3
$ cd build/

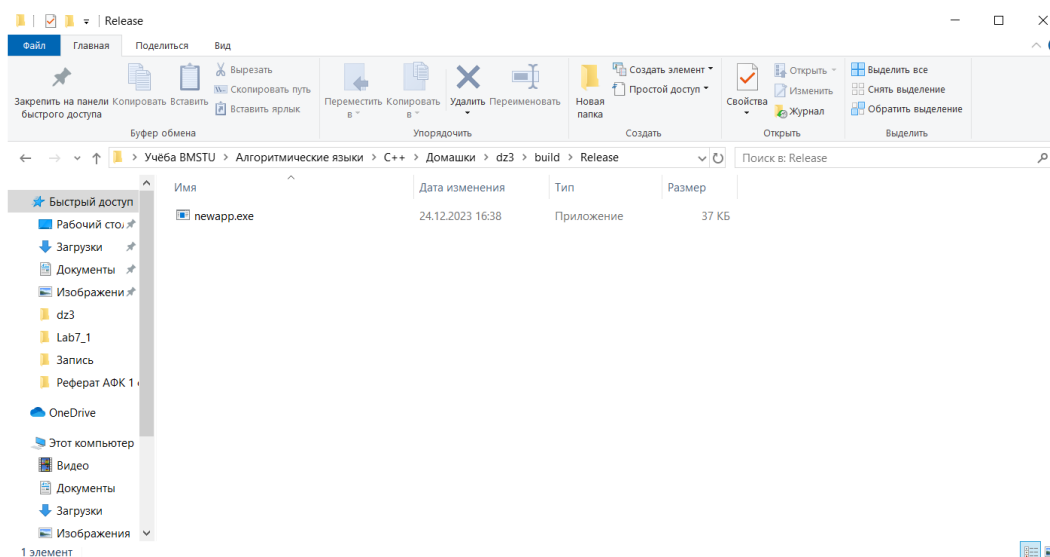
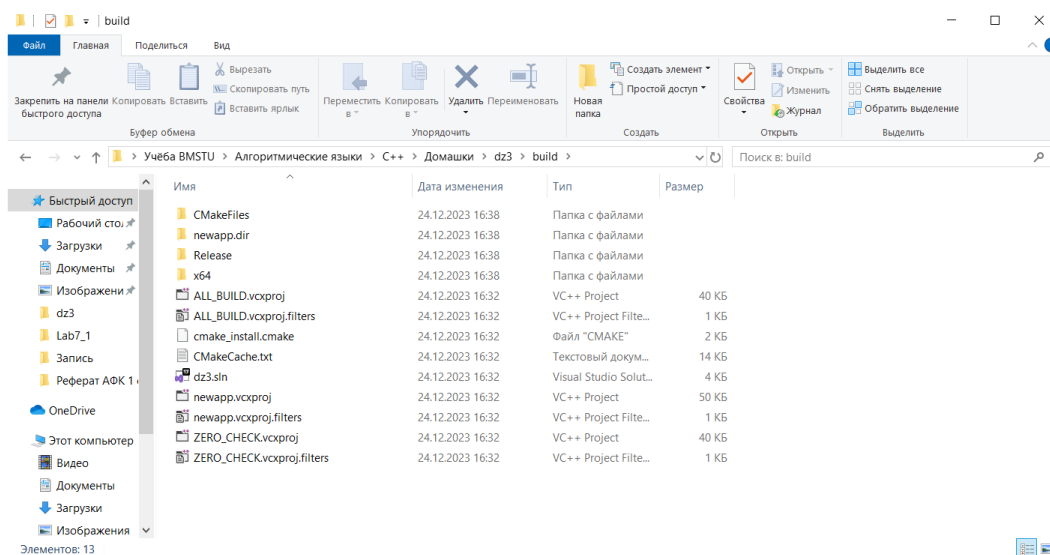
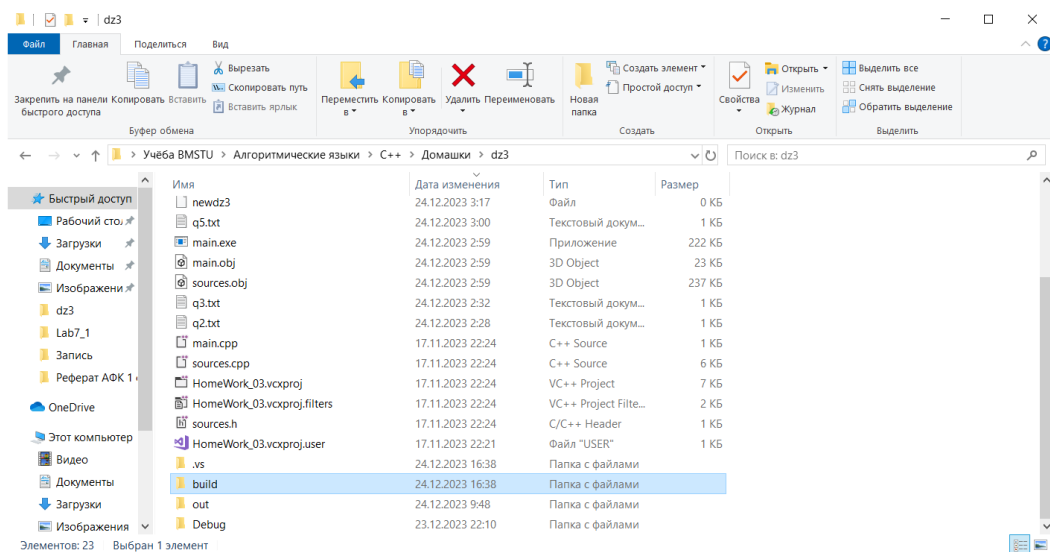
C:\Int000DESKTOP-7UTAD4U\MINGW64 ~/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/build
$ cmake ..
-- Building for: Visual Studio 17 2022
-- Selecting Windows SDK version 10.0.22621.0 to target Windows 10.0.19045.
-- The CXX compiler identification is MSVC 19.37.32822.0
-- Detecting CXX compiler ABI info
-- Detecting CXX compiler ABI info - done
-- Check for working CXX compiler: C:/Program Files/Microsoft Visual Studio/2022/Community/VC/Tools/MSVC/14.37.32822/bin/Hostx64/x64/cl.exe - skipped
-- Detecting CXX compile features
-- Detecting CXX compile features - done
-- Configuring done (1.8s)
-- Generating done (0.0s)
-- Build files have been written to: C:/Users/kwInt00/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/build

C:\Int000DESKTOP-7UTAD4U\MINGW64 ~/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/build
$ cmake --build; --config Release
Внимание! MSBuild 17.7.2-d6990bfa для .NET Framework

1>Checking Build System
Building Custom Rule C:/Users/kwInt00/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/CMakeLists.txt
main.cpp
sources.cpp
Создание кода...
C:/Users/kwInt00/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/sources.cpp(41): warning C4700: использована неинициализированная локальная переменная "consecutiveByte" [C:/Users/kwInt00/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/build/newapp.vcxproj]
newapp.vcxproj -> C:/Users/kwInt00/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/build/Release/newapp.exe
Building Custom Rule C:/Users/kwInt00/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/CMakeLists.txt

C:\Int000DESKTOP-7UTAD4U\MINGW64 ~/Desktop/Учеба БМСТУ/Алгоритмические языки/C++/Домашки/dz3/build
```

Рисунок 2 – сборка ПО с помощью СMake через командную строку



Рисунки 3-5 – файлы, созданные в процессе сборки через командную строку

Код программы:

Файл sources.cpp:

```
#include "sources.h"

void printBinaryFile(const std::string& sourceFileName)
{
    std::ifstream sourceFile(sourceFileName, std::ios::binary);

    if (!sourceFile)
    {
        std::cerr << "Failed to open file" << std::endl;
        return;
    }

    std::cout << "Binary representation of the file " << sourceFileName <<
std::endl;

    char currentByte;
    while (sourceFile.get(currentByte))
    {
        std::cout << std::hex << static_cast<int>(static_cast<unsigned
char>(currentByte)) << '\t';
    }
    std::cout << std::endl;

    sourceFile.close();
}

void compressFile(const std::string& sourceFileName, const std::string&
compressedFileName)
{
    std::ifstream sourceFile(sourceFileName, std::ios::binary);
    std::ofstream compressedFile(compressedFileName, std::ios::binary);

    if (!sourceFile || !compressedFile)
    {
        std::cerr << "Failed to open file" << std::endl;
        return;
    }

    char currentByte;
    char consecutiveByte;
    unsigned char consecutiveCount;
    while (sourceFile.get(currentByte))
    {
        if (currentByte == consecutiveByte)
        {
            consecutiveCount++;
        }
        else
        {
            if (consecutiveCount > 1)
            {
                unsigned char countByte = consecutiveCount + 0x80;
                compressedFile.write((char*)&countByte, sizeof(unsigned char));
                compressedFile.write(&consecutiveByte, sizeof(char));
            }
            if (consecutiveCount == 1)
            {
                compressedFile.write(&consecutiveByte, sizeof(consecutiveByte));
            }
            consecutiveByte = currentByte;
            consecutiveCount = 1;
        }
    }
}
```

```

    }
}

if (consecutiveCount > 1)
{
    unsigned char countByte = consecutiveCount + 0x80;
    compressedFile.write((char*)&countByte, sizeof(countByte));
    compressedFile.write(&consecutiveByte, sizeof(consecutiveByte));
}
if (consecutiveCount == 1)
{
    compressedFile.write(&consecutiveByte, sizeof(consecutiveByte));
}

sourceFile.close();
compressedFile.close();
}

void decompressFile(const std::string& compressedFileName, const std::string&
decompressedFileName)
{
    std::ifstream compressedFile(compressedFileName, std::ios::binary);
    std::ofstream decompressedFile(decompressedFileName, std::ios::binary);

    if (!compressedFile || !decompressedFile)
    {
        std::cerr << "Failed to open file" << std::endl;
        return;
    }

    char currentByte;
    while (compressedFile.get(currentByte))
    {
        if ((unsigned char)currentByte >= 0x80)
        {
            char count = (currentByte & 0x7F);
            char repeatedByte;
            if (compressedFile.get(repeatedByte))
            {
                for (size_t i = 0; i < count; ++i)
                {
                    decompressedFile.write(&repeatedByte, sizeof(repeatedByte));
                }
            }
        }
        else
        {
            decompressedFile.write(&currentByte, sizeof(currentByte));
        }
    }

    compressedFile.close();
    decompressedFile.close();
}

void convertBytes(const std::string& sourceFileName, const std::string&
convertedFileName)
{
    std::ifstream sourceFile(sourceFileName, std::ios::binary);
    std::ofstream convertedFile(convertedFileName, std::ios::binary);

    if (!sourceFile || !convertedFile)
    {
        std::cerr << "Failed to open file" << std::endl;
        return;
    }

```

```

    }

    char currentByte;
    while (sourceFile.get(currentByte))
    {
        currentByte = currentByte & 0x7F;
        convertedFile.write(&currentByte, sizeof(currentByte));
    }

    sourceFile.close();
    convertedFile.close();
}

void archiver(const std::string& mode, const std::string& sourceFileName, bool
printingBytes)
{
    if (mode == "-c")
    {
        std::string convertedFileName, compressedFileName;
        if (printingBytes)
        {
            printBinaryFile(sourceFileName);
        }
        std::cout << "Введите название преобразованного файла для архивации: ";
        std::cin >> convertedFileName;
        convertBytes(sourceFileName, convertedFileName);
        if (printingBytes)
        {
            printBinaryFile(convertedFileName);
        }
        std::cout << std::endl << "Введите название для архивированного файла: ";
        std::cin >> compressedFileName;
        compressFile(convertedFileName, compressedFileName);
        if (printingBytes)
        {
            printBinaryFile(compressedFileName);
        }
        std::cout << std::endl
            << "Successfully created " << compressedFileName << std::endl;
    }
    else
    {
        std::string decompressedFileName;
        if (printingBytes)
        {
            printBinaryFile(sourceFileName);
        }
        std::cout << std::endl << "Введите название для разархивированного файла: ";
        std::cin >> decompressedFileName;
        decompressFile(sourceFileName, decompressedFileName);
        if (printingBytes)
        {
            printBinaryFile(decompressedFileName);
        }
        std::cout << std::endl << "Successfully created " << decompressedFileName <<
std::endl;
    }
}

```

Файл sources.h:


```

#pragma once
#include <fstream>
#include <iostream>
#include <string>

void printBinaryFile(const std::string& sourceFileName);
void compressFile(const std::string& sourceFileName, const std::string&
compressedFileName);
void decompressFile(const std::string& compressedFileName, const std::string&
decompressedFileName);
void convertBytes(const std::string& sourceFileName, const std::string&
convertedFileName);

void archiver(const std::string& mode, const std::string& sourceFileName, bool
printingBytes = false);

```

Файл main.cpp:

```

#include "sources.h"

int main(int argc, char* argv[]) // argv[1] = -c compressFile /-d decompressFile,
argv[2] = sourceFileName
{
    setlocale(LC_ALL, "ru");

    std::string mode = argv[1], sourceFileName = argv[2];

    archiver(mode, sourceFileName);

    return 0;
}

```

Файл CMakeLists.txt:

```

cmake_minimum_required(VERSION 3.26.4)
set(CMAKE_CXX_STANDARD 17)
project(dz3 VERSION 1.0 LANGUAGES CXX)
add_executable(newapp main.cpp sources.cpp)

```

Примеры работы программы:

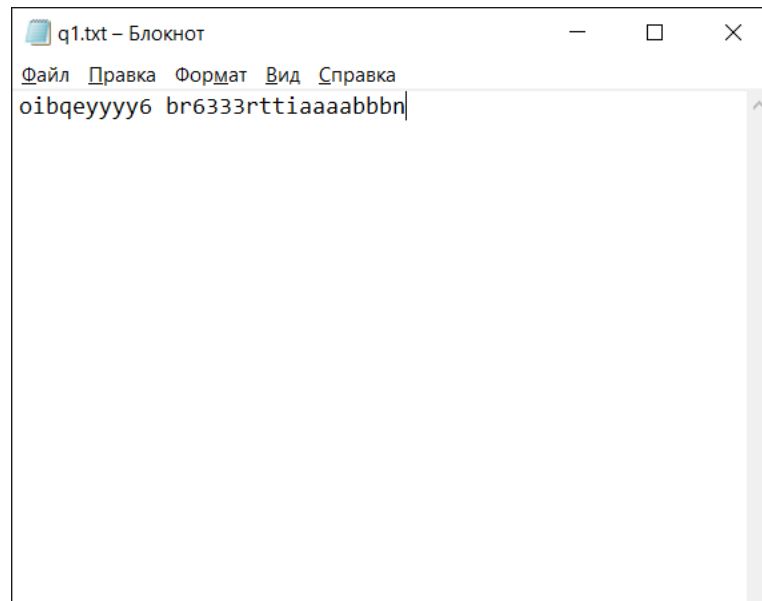


Рисунок 6 – исходный файл

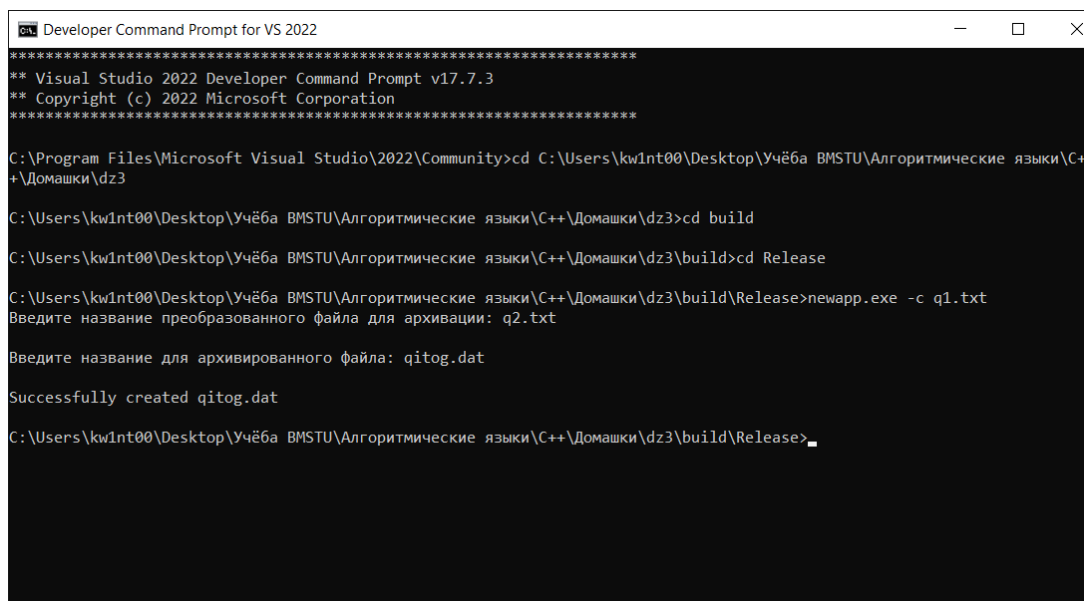


Рисунок 7 – запуск программы (сжатие)

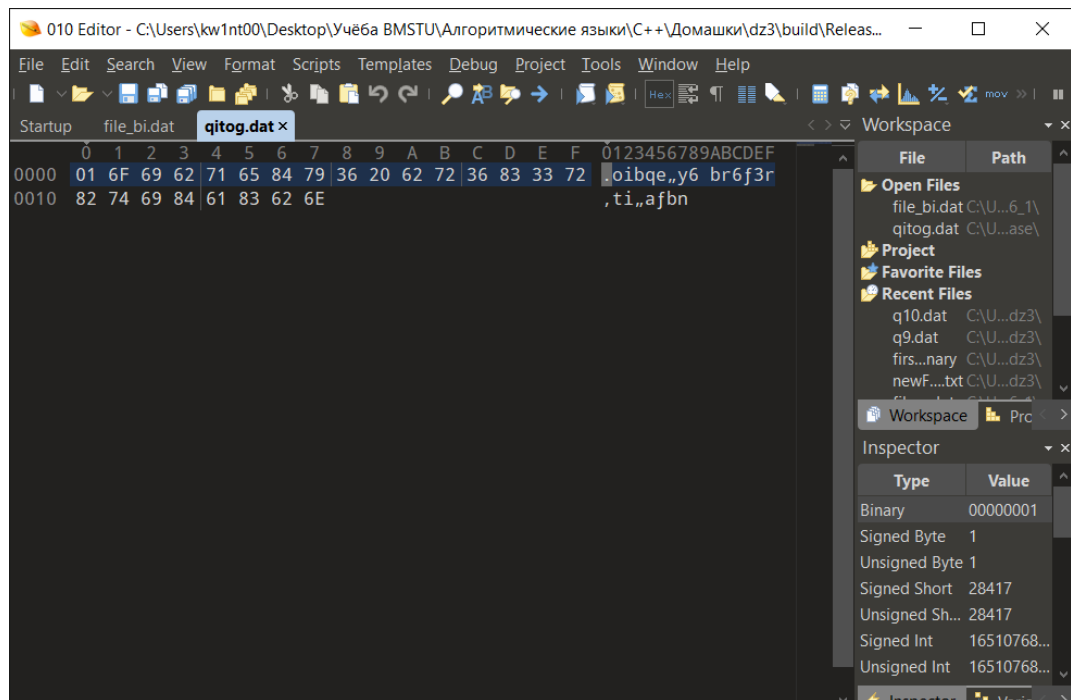


Рисунок 8 – сжатый файл (он же исходник для разархивирования)

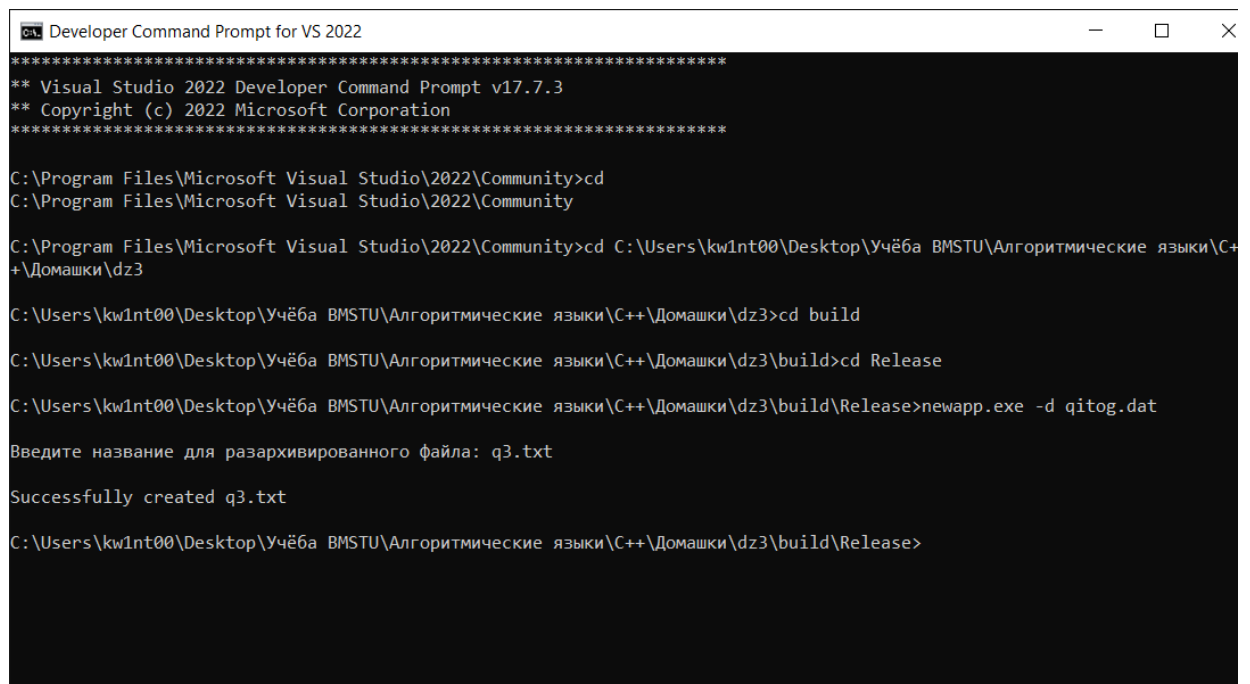


Рисунок 9 – запуск программы (разархивирование)

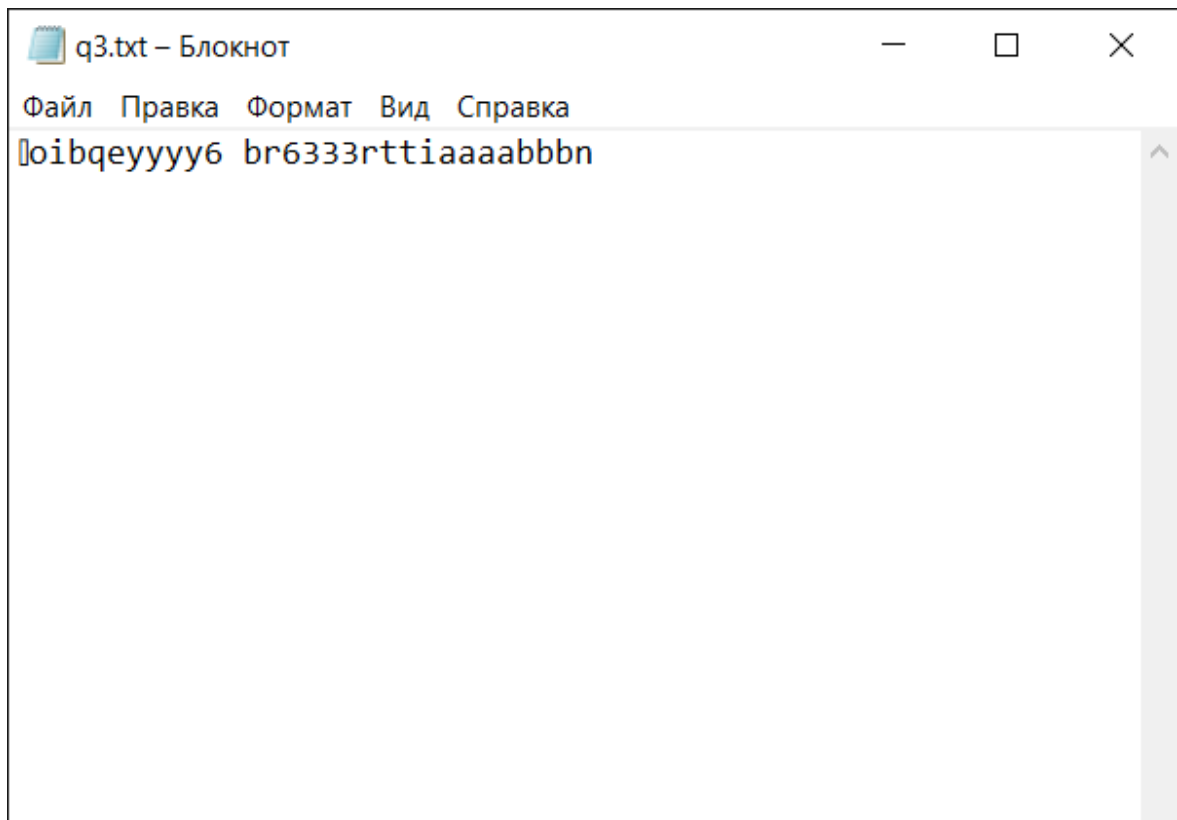


Рисунок 10 – разархивированный файл

Заключение

Задачи домашней работы были решены, результаты проверены. Разработана архитектура ПО, также реализована сборка ПО с помощью CMake через командную строку.