МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ федеральное государственное автономное образовательное учреждение высшего образования «САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

КАФЕДРА ВЫЧИСЛИТЕЛЬНЫХ СИСТЕМ И СЕТЕЙ

КУРСОВАЯ РАБОТА (ПРОЕКТ)
ЗАЩИЩЕНА С ОЦЕНКОЙ

РУКОВОДИТЕЛЬ

ассистент		Е.Е. Майн
должность, уч. степень, звание	подпись, дата	инициалы, фамилия

ПОЯСНИТЕЛЬНАЯ ЗАПИСКА К КУРСОВОЙ РАБОТЕ

по курсу: ИНФОРМАТИКА

РАБОТУ ВЫПОЛНИЛ

СТУДЕНТ ГР. № 4141		В.С. Сыворотнев
	подпись, дата	инициалы, фамилия

Санкт-Петербург 2022

МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное
образовательное учреждение высшего образования
«САНКТ-ПЕТЕРБУРГСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ АЭРОКОСМИЧЕСКОГО ПРИБОРОСТРОЕНИЯ»

Факультет 4

Кафедра 44

Задание по курсовому проект на тему:

Разработать музыкальный проигрыватель, для прослушивания аудиофайлов на Windows

1.СОДЕРЖАНИЕ ПРОЕКТА

Десктопное приложение с функционалом музыкального проигрывателя, позволяющее проигрывать *.mp3 файлы из папки в корне проекта на операционной системе Windows.

2.ПРОГРАММНЫЕ ИНСТРУМЕНТЫ РЕАЛИЗАЦИИ

1) CLion 2022.1 – язык программирования C++;

2) Комплект разработки программного обеспечения Microsoft Windows SDK

3.ОФОРМЛЕНИЕ ПРОЕКТА

3.1 Пояснительная записка, оформления в соответствии требованиям ГОСТ 2.105-2019 – ЕСКД. Общие требования к текстовым документам и ГОСТ 19.701-90 ЕСПД. Схемы алгоритмов, программ, данных и систем. Обозначения условные и правила выполнения. Загружается в личный кабинет в формате *.pdf.
3.2 Формат презентации: блок-схема программы, текст программы, работоспособность программы, результаты тестирования программы.

4.УКАЗАНИЯ

4.1 Исходя из рекомендованной структуры курсовой работы, ее объем должен составлять примерно 20-25 страниц, которые на ряду с описанием программы, включают способ взаимодействия с программой, описание спецификации функций, возможность выбора варианта действия с помощью меню. Обеспечить показ на экране меню взаимодействия с программой, скриншоты контрольных примеров, используемых при проверке работоспособности курсовой работы.

5.ЛИТЕРАТУРА

5.1 Документация по Microsoft: C++ https://docs.microsoft.com/ru-ru/cpp/ 17.03.2022

5.2 Win SDK Documentation: https://docs.microsoft.com/ru-ru/cpp/windows/how-to-use-the-windows-10-sdk-in-a-windows-desktop-application?view=msvc-170 17.03.2022

5.3 Руководство по использованию среды разработки Clion - jetbrains.com/ru-ru/clion/learn/ 17.03.2022

Руководитель курсовой работы		Е.Е. Майн
	дата, подпись	расшифровка
Задание принял к исполнению		В.С. Сыворотнев
	дата, подпись	расшифровка

СОДЕРЖАНИЕ

ТЕРМИНЫ И ОПРЕДЕЛЕНИЯ

Блок-схема — распространённый тип схем, описывающих алгоритмы или процессы, в которых отдельные шаги изображаются в виде блоков различной формы, соединённых между собой линиями, указывающими направление последовательности.

Музыкальный плеер — компьютерная программа, предназначенная для воспроизведения файлов мультимедиа-содержимого.

Трек — песня, музыкальная композиция или другой отдельный отрезок звукозаписи.

CLion — интегрированная среда разработки для языков программирования Си и C++, представляющая собой многофункциональную программу, облегчающую написание кода и ускоряющую этот процесс.

ВВЕДЕНИЕ

Данная работа направлена на реализацию прикладного музыкального плеера, который может быть использован в повседневной жизни, а также на закрепление умений и навыков, приобретенных в ходе изучения курса «Информатика».

Цель курсового проекта – решение задачи, сформулированной в индивидуальном задании: разработать музыкальный проигрыватель, для прослушивания аудиофайлов на Windows.

Помимо этого, представленная работа несет в себе задачу приобретения навыков оформления курсовых работ.

1.ПОСТАНОВКА ЗАДАЧИ

1.1 . Цели и задачи

Целью курсовой работы является написание музыкального плеера на платформе Windows.

Для достижения цели будут выполнены следующие задачи:

- изучение структурной организации объектно-ориентированной архитектуры программы, многопоточной декомпозиции задач, обработки файлов, работы с внешними и встроенными библиотеками на языке программирования C++;

- построение схемы алгоритма решения задачи;

- разработка программы при помощи использования технологии ПП и ООП;

-оформление курсовой работы на тему «разработать музыкальный проигрыватель, для прослушивания аудиофайлов на Windows.».

1.2. Выбор средства разработки

Выбор CLion в качестве среды программирования был обусловлен личным удобством использования продуктов компании JetBrains, а также доступностью бесплатной подписки на продукт для студентов.

В качестве основного языка программирования были рассмотрены C++ и C#. C++ был выбран по причине большего количества библиотек для реализации поставленной задачи, возможностью более гибкого управления оперативной памятью, гибкой сборки и более высокой производительности. Результат сравнения языков показан в Таблице 1

Таблица 1 – Сравнение языков программирования

Категория сравнения	C++	C#
Функциональность	Более гибкое управление памятью, большее кол-во	Автоматическое управление памятью, является более

	подключаемых библиотек	высокоуровневым, меньше библиотек
Производительность	Результаты тестирования при работе выводятся быстрее, чем в C#	Работает медленнее, чем программа-оппонент.
Сборка	Более сложная и гибкая возможность сборки приложений	Упрощенная сборка

Таблица 1 – Сравнение языков программирования

Для решения поставленной была использована библиотека Windows SDK. У этой библиотеки наиболее подробная документация с примерами работы на языке программирования C++, а также достаточно простое взаимодействие с инструментами воспроизведения музыкальных файлов в операционной системе Windows.

2. РАЗРАБОТКА БЛОК СХЕМ

Работа программы начинается с вывода доступных треков. Пользователю предлагается ввести номер выбранного трека. Далее начинается его воспроизведение. Пользователь сможет выйти в меню, поставив трек на паузу, введя 0. Далее можно выбрать другие функции плеера в меню. Блок-схема алгоритма главной части программы представлена на рисунке 1.

Класс аудиоплеер имеет публичных 12 методов. CallMciSendString упрощает использование метода msiSendString, уменьшая количество принимаемых аргументов, и делая код чище и визуально приятней. GetAvailableTracks позволяет получить названия треков из папки music в корне проекта, возвращает массив названий. CloseAudio метод, заканчивающий работу с определенным треком. PlayAudio метод, начинающий воспроизведение трека. ChangeVolume изменяет громкость проигрываемого трека. RewindAudio перематывает трек на выбранный промежуток. ChangeTrack производит смену трека. RestarAudio запускает трек сначала. PauseAudio ставит трек на паузу. IsPlaying возвращает значение приватного поля isPlaying. GetVolumeValue возвращает значение приватного поля currentVolume. IsTrackChosen возвращает значение приватного поля selectedTrackName.

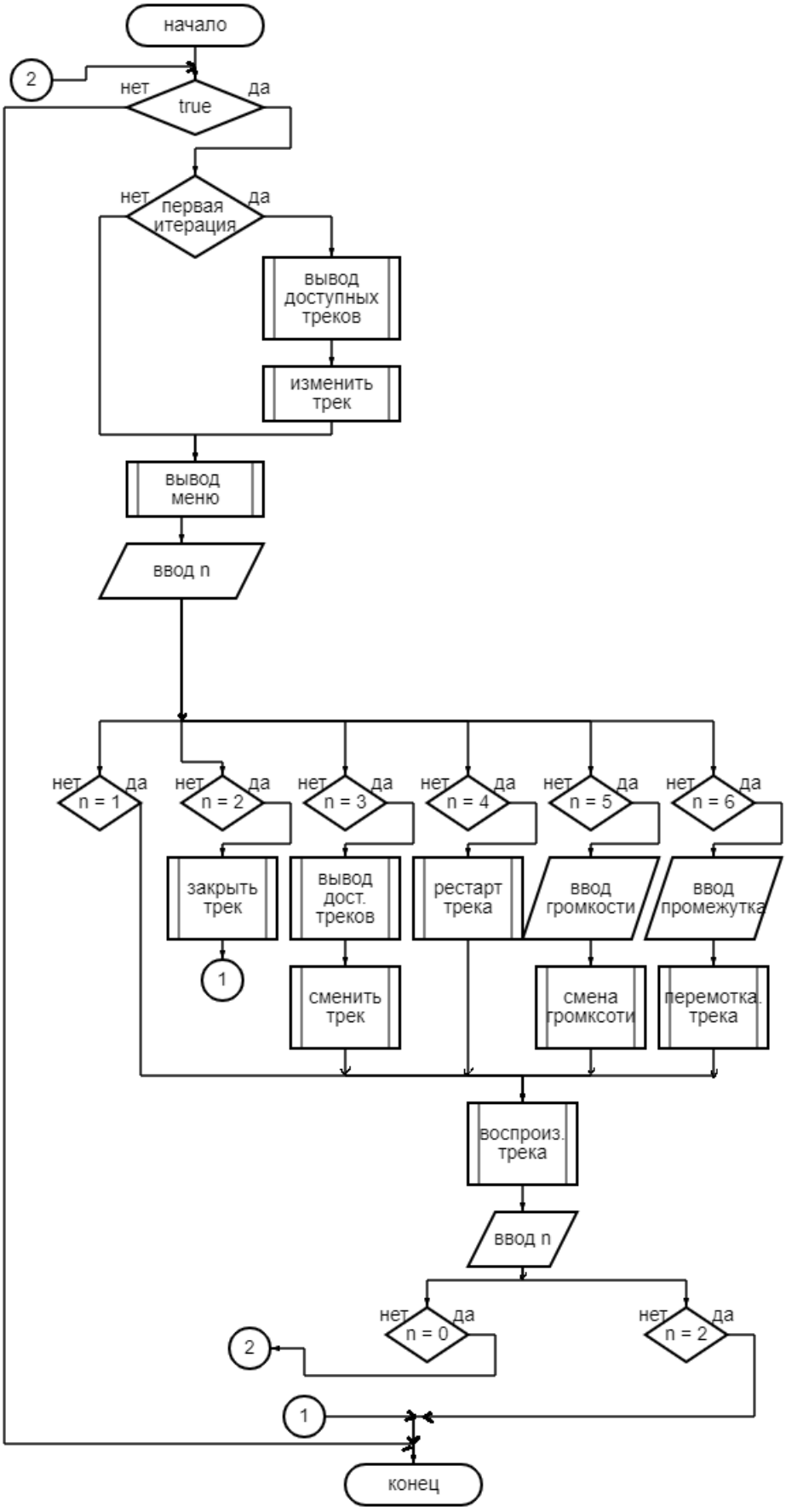

Рисунок 1 – Блок-схема основного алгоритма работы программы

3. РАЗРАБОТКА ПРОГРАММЫ

3.1. Описание программы

Программа написана на языке C++. Программа решает задачу, сформулированную в индивидуальном задании: «разработать музыкальный проигрыватель, для прослушивания аудиофайлов на Windows.».

3.1.1. Описание логической структуры

При запуске программы открывается консольное меню, в котором пользователю перечисляются пронумерованные аудиофайлы, находящиеся в папке “/music” в корне проекта, и пользователю предлагается ввести номер желаемого аудиофайла для начала его прослушивания. После ввода значения воспроизведение трека начинается автоматически, и пользователю выводится сообщения о выборе дальнейшего действия: для того, чтобы поставить аудиофайл на паузу, необходимо ввести 0, а для выхода из программы ввести 2. При этом пользователь так же получает сообщение, что для того, чтобы получить доступ к другим функциям аудиоплеера (сменить трек, перемотать его, изменить громкость, начать его сначала) необходимо поставить аудиофайл на паузу. После того, как пользователь ставить композицию на паузу, он сразу же получает меню, в котором есть на выбор 6 действий: продолжить воспроизведение (Цифра 1), выйти из программы (Цифра 2), изменить трек (Цифра 3), начать трек заново(Цифра 4), изменить громкость (Цифра 5), перемотать трек (Цифра 6).

3.2 Текст программы

Файл main.cpp

```
#include "audioplayer.h"

void PrintMenu();

void PrintAvailableTracks(const AudioPlayer& audioPlayer);
```

```
int main() {
  AudioPlayer audioPlayer;
  bool firstIteration = true;
  SetConsoleOutputCP(CP_UTF8);

  // главный цикл для работы программы
  while (true) {
    int n;

    if (!firstIteration) {
      // вывод меню
      PrintMenu();

      // считываение цифрового значения, для выбора действия
      cin >> n;

      // проверка на корректность введенного значения
      if (isdigit(n) == n) {
        cout << "\nОшибка. Введите корректное значение";
        return 1;
      }

    } else {
      firstIteration = false;
      int trackNumber;

      // вывод доступных аудиофайлов, находящихся в папке music в корне
проекта
      PrintAvailableTracks(audioPlayer);
```

```
  // ввод номера трека, который пользователь хочет включить
  cout << "\nЧтобы выбрать трек, введите его номер." << endl;
  cin >> trackNumber;

  // вызов метода смены трека класса Audioplayer
  audioPlayer.ChangeTrack(trackNumber);
}

// если пользователь ввел 1, то воспроизводим трек
if (n == 1) {
  if (!audioPlayer.IsTrackChosen()) {
    cout << "Невозможно найти трек" << endl;
    break;
  }
  audioPlayer.PlayAudio();

  // если пользователь ввел 2, то выходим из программы
} else if (n == 2) {
  audioPlayer.CloseAudio();
  break;

  // если пользователь ввел 3, то производим смену трека
} else if (n == 3) {
  int trackNumber;

  PrintAvailableTracks(audioPlayer);

  cout << "\nЧтобы выбрать трек, введите его номер." << endl;
  cin >> trackNumber;
  if (!audioPlayer.ChangeTrack(trackNumber)) {
```

```
      cout << "Ошибка смены трека";
      break;
    }

    // если пользователь ввел 4, то перезапускаем трек
  } else if (n == 4 && !audioPlayer.IsPlaying()) {

    if (!audioPlayer.IsTrackChosen()) {
      cout << "Невозможно найти трек";
      break;
    }
    audioPlayer.RestartAudio();

    // если пользователь ввел 5, то изменяем громкость на выбранную
  } else if (n == 5 && !audioPlayer.IsPlaying()) {
    int volume;

    if (!audioPlayer.IsTrackChosen()) {
      cout << "Невозможно найти трек";
      break;
    }

    cout << "\nТекущий уроквень громкости: " <<
audioPlayer.GetVolumeValue();
    cout << "\nЧтобы установить уровень громкости, введите значение
(минимум - 0; максимум - 1000)" << endl;
    cin >> volume;

    if (!audioPlayer.ChangeVolume(volume)) {
      cout << "Ошибка изменения громкости";
```

```
        break;
      }

      // если пользователь ввел 6, то позволяем ему перемотать трек на выбранный промежуток
    } else if (n == 6) {
      string trackTiming;

      cout << "Введите секунду, на которую вы хотите перемотать трек: " << endl;
      cin >> trackTiming;

      if (!audioPlayer.IsTrackChosen()) {
        cout << "Невозможно найти трек";
        break;
      }
      audioPlayer.RewindAudio(trackTiming);
    }

    // внутреннее меню, если трек возпроизводится
    if (audioPlayer.IsPlaying()) {
      cout << "Введите 0 чтобы поставить файл на паузу или введите 2 чтобы выйти из файла." << endl;
      cout << "Чтобы сменить трек, перемотать его, изменить громкость или начать сначала, нажмите паузу" << endl;
      cin >> n;

      // если пользователь ввел 0 во время воспроизведения аудиотрека, то ставим трек на паузу
      if (n == 0) {
```

```
        //pause the audio file
        audioPlayer.PauseAudio();

      // если пользователь ввел 2 во время воспроизведения аудиотрека, то выходим из программы
      } else if (n == 2) {
        //close the audio file
        audioPlayer.CloseAudio();
        break;
      }
    }
  }
}

// функция, производящая вывод консольного меню
void PrintMenu() {
  cout << "----------------------------" << endl;
  cout << "-Введите 1 чтобы воспроизвести файл" << endl;
  cout << "-Введите 2 чтобы выйти из программы" << endl;
  cout << "-Введите 3 чтобы изменить файл" << endl;
  cout << "-Введите 4 чтобы воспроизвести файл сначала" << endl;
  cout << "-Введите 5 чтобы изменить громкость" << endl;
  cout << "-Введите 6 чтобы перемотать трек" << endl;
  cout << "----------------------------" << endl;
}

// функция, производящая вывод доступных треков в папке music в корне проекта
void PrintAvailableTracks(const AudioPlayer& audioPlayer) {
  int counter = 0;
```

```
  const auto files = audioPlayer.GetAvailableTracks("music/");

  cout << "\nфайлы в папке 'music':" << endl;
  for (auto const file: files) {
    counter++;
    cout << counter << ". " << file << endl;
  }
}
```

Файл audioplayer.cpp

```
#include "audioplayer.h"

// метод класса Audioplayer, который позволяет получить названия треков из папки music в корне проекта. Возвращает массив названий
vector<string> AudioPlayer::GetAvailableTracks(const string& folder) const {
  vector<string> names;
  DIR* directory = opendir(folder.c_str());

  // проверка на валидность наличия папки внутри проекта
  if (directory == nullptr) {
    throw invalid_argument("Unable to find directory");
  }

  dirent* ent;
  while ((ent = readdir(directory)) != nullptr) {
    if ((strcmp(ent->d_name, "..")) && (strcmp(ent->d_name, "."))) {
      names.emplace_back(ent->d_name);
    }
  }
  closedir(directory);
```

```
  return names;
}

// метод класса AudioPlayer, который заканчивает работу с определенным треком
void AudioPlayer::CloseAudio() const {
  CallMciSendString("close mp3");
}

// метод класса AudioPlayer, который начинает воспроизведение трека
void AudioPlayer::PlayAudio() {
  // сохраняем текущую громкость
  CallMciSendString("setAudio mp3 volume to " + to_string(currentVolume));

  isPlaying = true;
  CallMciSendString("play mp3");
}

// метод класса AudioPlayer, который изменяет громкость проигрываемого трека
bool AudioPlayer::ChangeVolume(const int volume) {
  if (volume > 1000 || volume < 0) {
    return false;
  }

  currentVolume = volume;
  CallMciSendString("setAudio mp3 volume to " + to_string(currentVolume));

  return true;
}
```

```
// метод класса AudioPlayer, который перматывает трек на определенный промежуток
void AudioPlayer::RewindAudio(const string& trackTiming) {
  // rewind
  CallMciSendString(("seek mp3 to " + trackTiming + "000"));

  PlayAudio();
}

// метод класса AudioPlayer, который производит смену трека
bool AudioPlayer::ChangeTrack(const int trackNumber) {
  vector<string> files = GetAvailableTracks("music/");

  if (trackNumber < 0 || trackNumber > files.size()) {
    return false;
  }

  // закрываем предыдущий трек
  CallMciSendString("close mp3");

  selectedTrackName = files[trackNumber - 1];
  CallMciSendString(+"open \"music/" + selectedTrackName + "\"" + " type mpegVideo alias mp3");

  PlayAudio();

  return true;
}
```

```
// метод класса AudioPlayer, который позволяет запустить трек снова
void AudioPlayer::RestartAudio() {
  CallMciSendString("seek mp3 to start");

  PlayAudio();
}

// метод класса AudioPlayer, который ставит трек на паузу
void AudioPlayer::PauseAudio() {
  //pause the audio file
  isPlaying = false;
  CallMciSendString("pause mp3");
}

// метод класса AudioPlayer, который упрощает использование метода
msiSendString, уменьшая колличество принимаемых
// аргументов, и делая код чище и визуально приятней
void AudioPlayer::CallMciSendString(const string& str) const {
  mciSendString(str.c_str(), nullptr, 0, nullptr);
}

// метод класса AudioPlayer, который возвращает значение булевой
переменной isPlaying, которая показывает состояние трека
// (играет - true, не играет - false)
// этот метод необходим для того, чтобы поля класса оставались приватными
bool AudioPlayer::IsPlaying() const {
  return isPlaying;
}

// метод класса AudioPlayer, который возвращает значение числовой
```

переменной currentVolume, в которой хранится текущее значение громкости
// этот метод необходим для того, чтобы поля класса оставались приватными
int AudioPlayer::GetVolumeValue() const {
  return currentVolume;
}

// метод класса AudioPlayer, который возвращает false, если трек не выбран, или true, если трек выбран
bool AudioPlayer::IsTrackChosen() const {
  return selectedTrackName != "";
}

Файл audioplayer.h

```
        #pragma once
#include <Windows.h>
#include <MMSystem.h>
#include <iostream>
#include <io.h>
#include <dirent.h>
#include <vector>
#include <string>
#include <algorithm>

#pragma comment(lib, "winMM.lib")

using namespace std;

class AudioPlayer {
 private:
  string selectedTrackName;
```

```
 int currentVolume = 50;
 bool isPlaying = false;

public:
 void CallMciSendString(const string& str) const;
 vector<string> GetAvailableTracks(const string& folder) const;
 bool ChangeVolume(const int volume);
 void RewindAudio(const string& trackTiming);
 void CloseAudio() const;
 bool ChangeTrack(const int trackNumber);
 void RestartAudio();
 void PlayAudio();
 void PauseAudio();
 bool IsPlaying() const;
 int GetVolumeValue() const;
 bool IsTrackChosen() const;
};
```

3.3 Результаты тестирования

Результат работы программы представлен на рисунке 2.

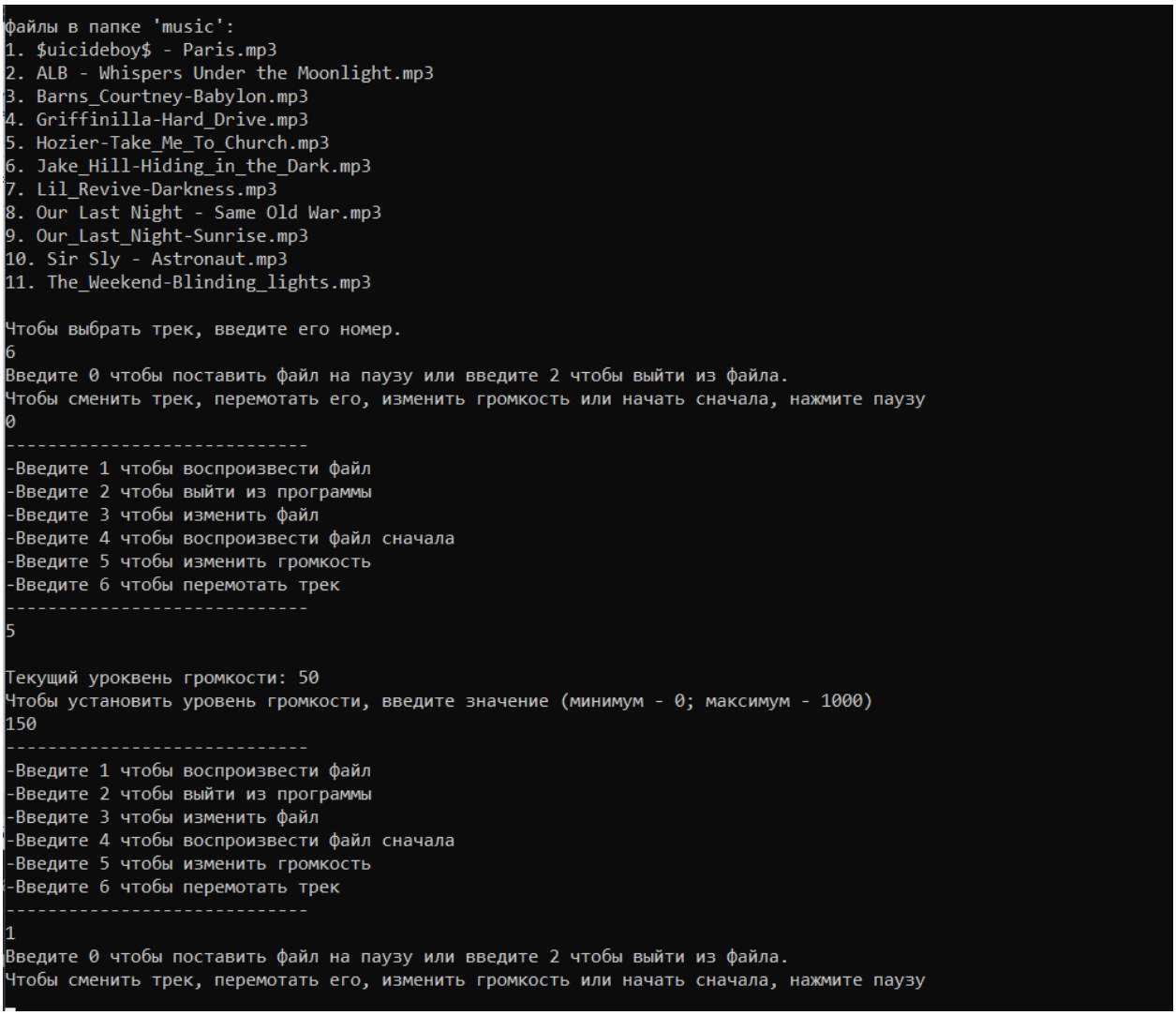

Рисунок 2 – Результат выполнения программы

4. ПРОВЕРКА РАБОТОСПОСОБНОСТИ ПРОГРАММЫ

На рисунке 3 показано начало работы программы

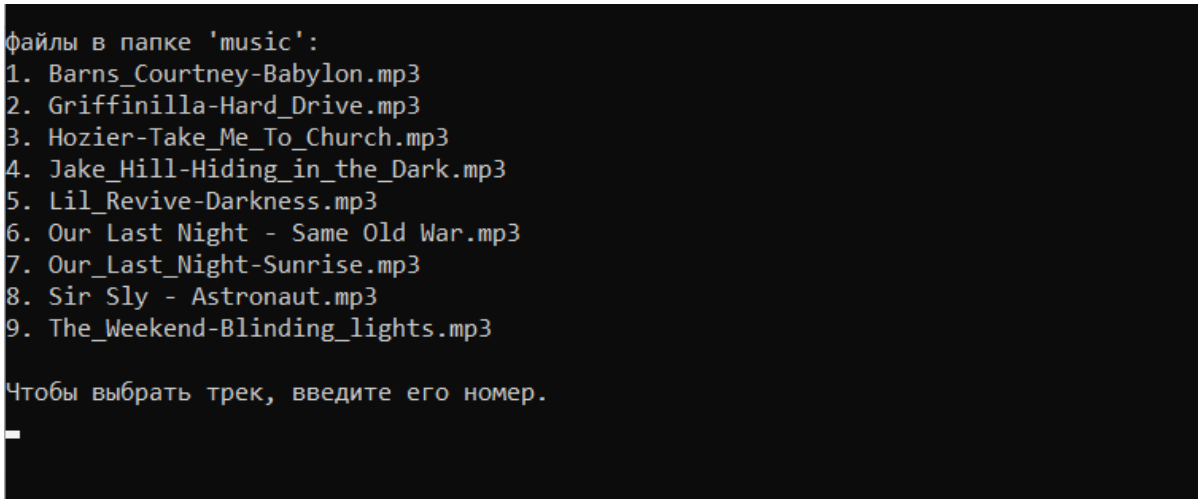

Рисунок 3 – Начало работы

На рисунке 4 показано состояние программы после выбора аудиофайла

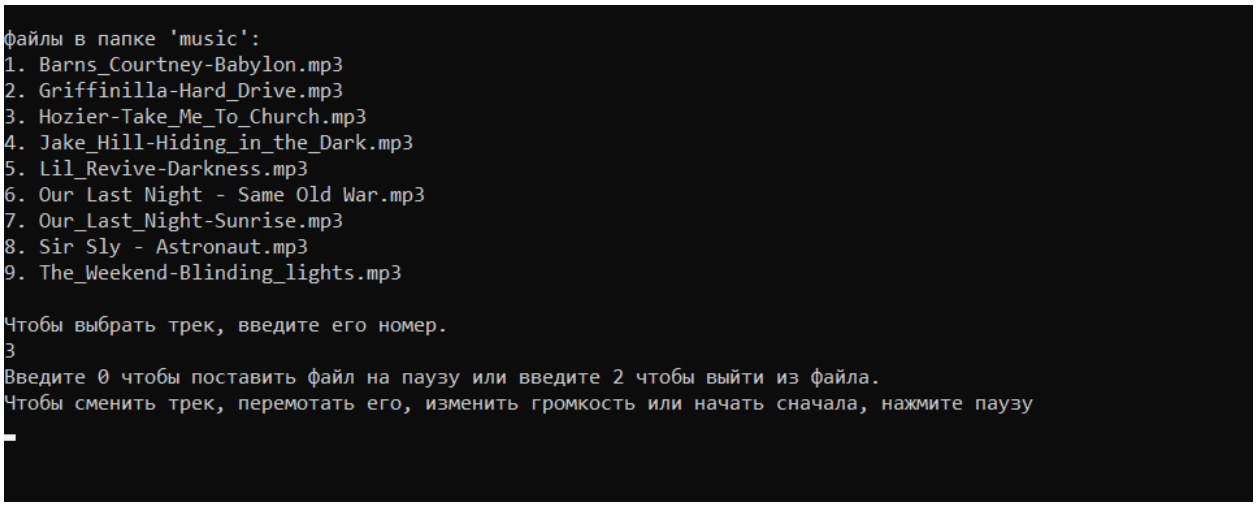

Рисунок 4 – Выбор желаемого аудиофайла

На рисунке 5 показано меню для работы с аудиоплеером

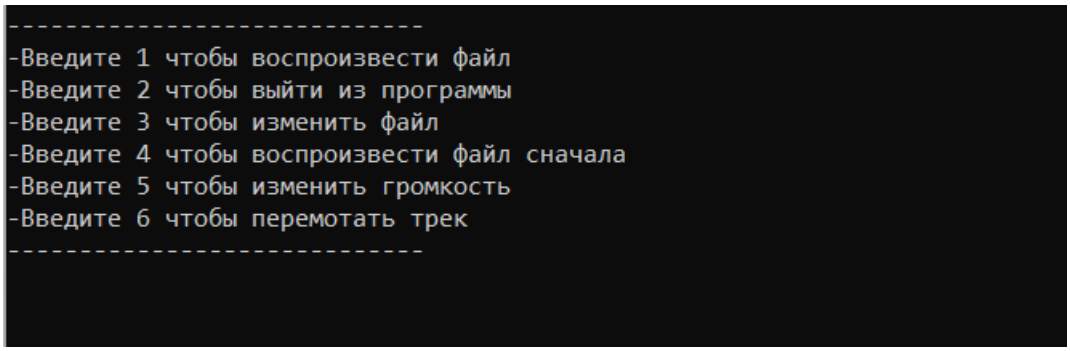

Рисунок 5 – Меню

На рисунке 6 показана функция смены аудиофайла

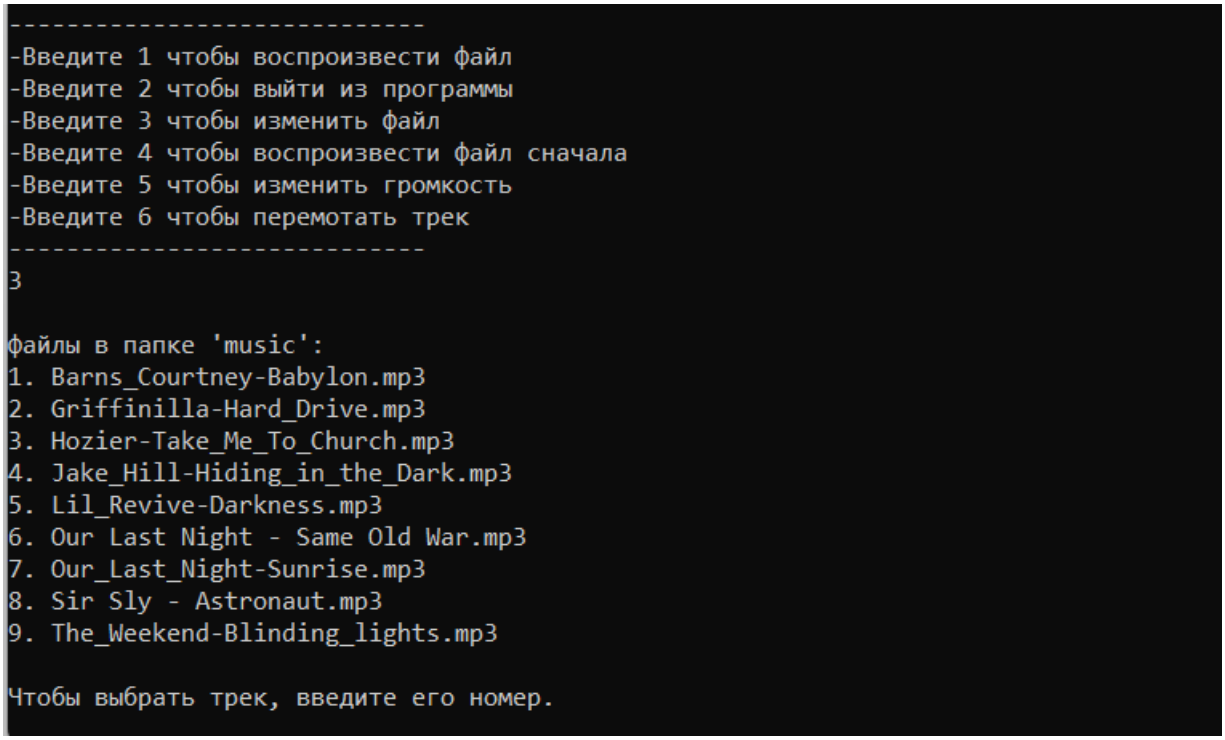

Рисунок 6 – Изменение аудиофайла

На рисунке 7 показана функция начала аудиофайла сначала

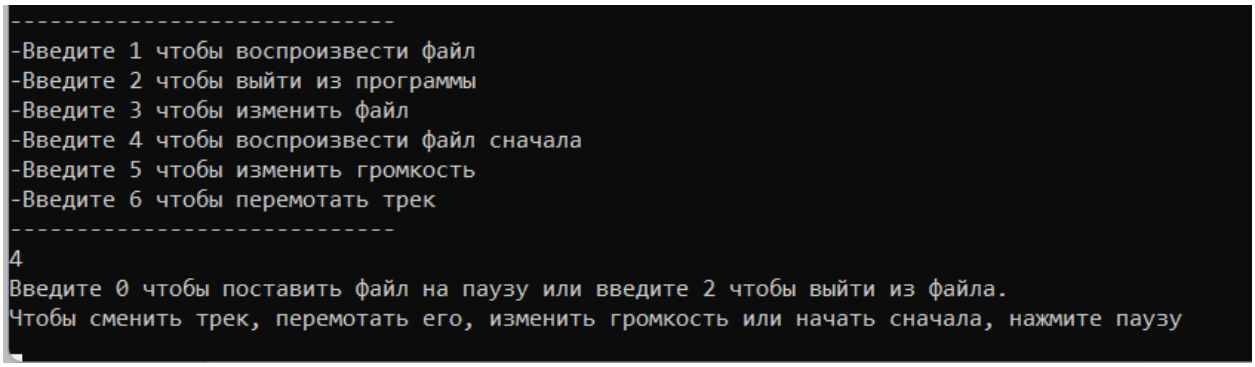

Рисунок 7 – Начало аудиофайла сначала

На рисунке 8 показана функция изменения громкости аудиофайла

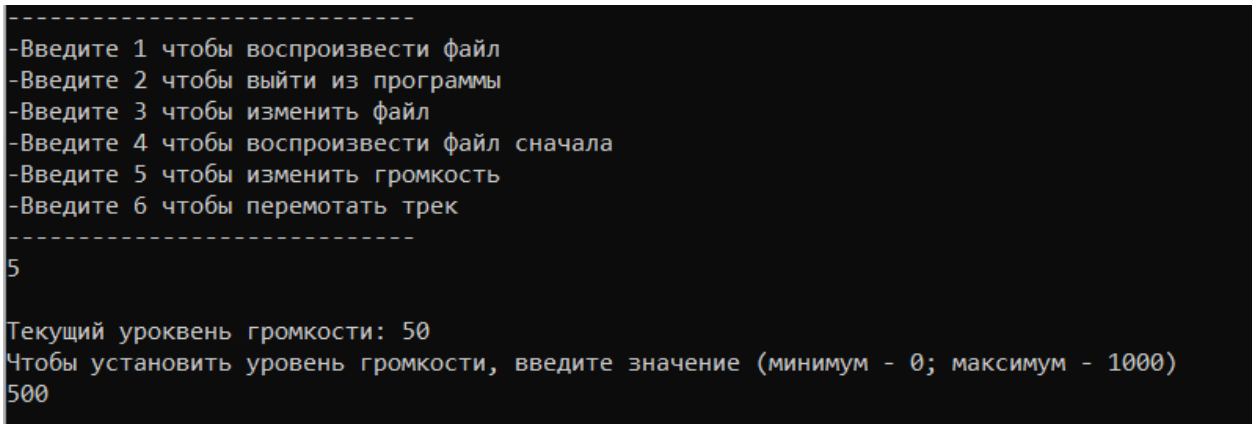

Рисунок 8 – Изменение громкости

На рисунке 9 показана функция перемотки аудиофайла

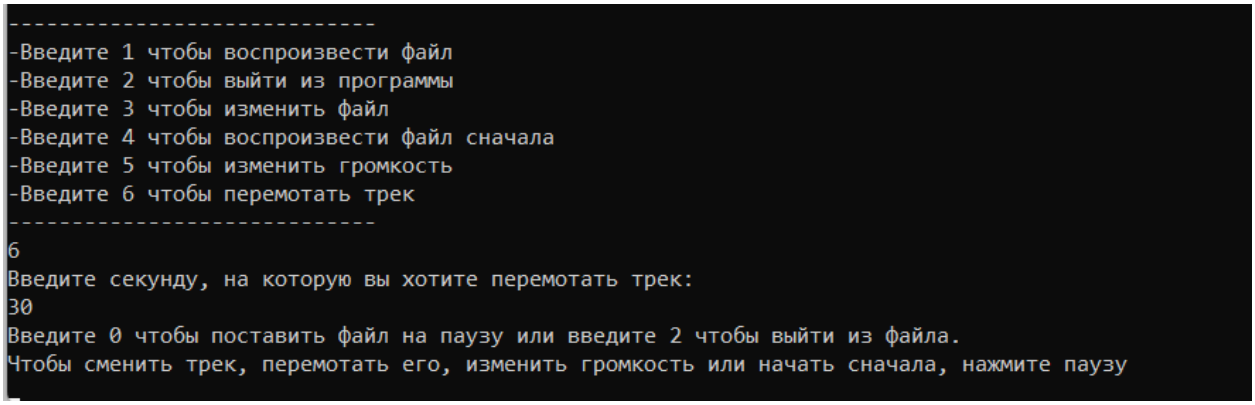

Рисунок 9 – Перемотка аудиофайла

ЗАКЛЮЧЕНИЕ

В данной работе была описана программа, позволяющая воспроизводить музыкальные файлы на операционной системе Windows.

Программа позволяет воспроизводить треки, менять их не прекращая работы, перематывать их, настраивать громкость, воспроизводить сначала, ставить на паузу. При написании данной работы были усовершенствованы навыки ПП и ОПП на языкеC++, а также освоена структура написания и оформления курсовых работ. Задачи были выполнены, цель работы достигнута.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Документация по Microsoft: [Электронный доступ], URL - https://docs.microsoft.com/ru-ru/cpp/, (режим доступа свободный).

2.ГОСТ 19.701-90 – ЕСПД Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.

3.ГОСТ 2.105-2019 – ЕСКД. Общие требования к текстовым документам,

4. Документация по Microsoft [Электронный доступ], URL - https://docs.microsoft.com/ru-ru/cpp/, (режим доступа свободный).

5.Руководство по использованию среды разработки Clion [Электронный доступ], URL - jetbrains.com/ru-ru/clion/learn/, (режим доступа свободный).