УЧЕБНО-НАУЧНО-ПРОИЗВОДСТВЕННЫЙ КОМПЛЕКС
«МЕЖДУНАРОДНЫЙ УНИВЕРСИТЕТ КЫРГЫЗСТАНА»

«УТВЕРЖДЕНО»
Ректор НОУ УНПК «МУК»
к.т.н., доцент Савченков Е.Ю.
_____
«16» 10 2018 г.

МАГИСТРАТУРА
Кафедра «**Компьютерных информационных систем и управления**»
Учебно-методический комплекс дисциплины
Алгоритмические основы мультимедийных технологий

Направление: **710100 «Информатика и вычислительная техника»**
Профиль: **Компьютерные информационные системы**
Академическая степень – **магистр**
Форма обучения – **очная**

График проведения модулей 1, 2 - семестры

неделя	**1**	**2**	**3**	**4**	**5**	**6**	**7**	**8 Мод.**	**9**	**10**	**11**	**12**	**13**	**14**	**15 Мод.**	**16**
лекц. зан.	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
лаб. зан.	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2	2

«РАССМОТРЕНО»
Протокол заседания кафедры
«КИСиУ»
№ 2 от 15/10.2018
Зав. кафедрой д.т.н., проф. Миркин Е.Л.
_____

«СОГЛАСОВАНО»
Проректор по академ. вопросам
проф. Мадалиев М.М.
_____

Составитель _____ к.т.н., и. о. доцент Нежинских С.С.

Директор Научной библиотеки _____ Асанова Ж.Ш.

БИШКЕК 2018

ОГЛАВЛЕНИЕ

Аннотация
Учебно-методический комплекс дисциплины (модулей)
1. Пояснительная записка
1.1 . Миссия и Стратегия
1.2 . Цель и задачи дисциплины (модулей)
1.3 . Формируемые компетенции, а также перечень планируемых (ожидаемых) результатов обучения по дисциплине (модулю) (знания, умения владения), сформулированные в компетентностном формате
1.4 . Место дисциплины (модулей) в структуре основной образовательной программы
2. Структура дисциплины (модулей)
3. Содержание дисциплины (модулей)
4. Конспект лекций
5. Информационные и образовательные технологии
6.Фонд оценочных средств для текущего, рубежного и итогового контролей по итогам освоения дисциплины (модулей)
6.1. Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины
6.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности
6.3. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания
6.4. Контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности
7. Учебно-методическое и информационное обеспечение дисциплины
7.1.Список источников и литературы
7.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей)
8. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся
8.1. Планы практических (семинарских) и лабораторных занятий. Методические указания по организации и проведению
8.2. Методические указания для обучающихся, по освоению дисциплины (модулей)
8.3. Методические рекомендации по подготовке письменных работ

9. Материально-техническое обеспечение дисциплины (модулей)
10. Приложения

АННОТАЦИЯ

Цель дисциплины (модулей) заключается в подготовке выпускника, имеющего специальные знания в области информационных технологий, для работы в области разработки мультимедийных систем. Задачами дисциплины (модулей) являются: получение знаний о современных методах кодирования мультимедийных данных; изучение кодирования и сжатия информации; получение навыков использования прикладных мультимедийных библиотек; решение типовых задач обработки мультимедийных данных. На изучение дисциплины отводится 90 часов. Основные разделы дисциплины: современные методы кодирования мультимедийных данных, кодирование и сжатие информации, прикладные мультимедийные библиотеки. Рубежный контроль успеваемости проводится на 5, 10, 15 неделях. Формы текущего контроля: опрос, проверка задания, посещаемость. Форма рубежного контроля — модульная работа. Форма итогового контроля — экзамен.

УЧЕБНО-МЕТОДИЧЕСКИЙ КОМПЛЕКС ДИСЦИПЛИНЫ (МОДУЛЕЙ)

1. Пояснительная записка

1.1. Миссия и стратегия

Подготовка профессионалов в своей будущей деятельности путем создания новых знаний и умений, способствование сохранению, приумножению научных, культурных и нравственных ценностей общества. Активизация разработки и внедрения новых организационных форм и методов обучения, максимально мотивирующих активную творческую работу, как обучающихся, так и преподавателей.

Стратегии развития - модернизация образовательной деятельности университета – совершенствование образовательного процесса в соответствии с требованиями Болонского процесса.

1.2. Цель и задачи дисциплины (модулей)

Цель дисциплины: подготовка выпускника, имеющего специальные знания в области информационных технологий, для работы в области разработки мультимедийных систем.

Задачи дисциплины:

- получение знаний о современных методах кодирования мультимедийных данных;
- изучение кодирования и сжатия информации;
- получение навыков использования прикладных мультимедийных библиотек;
- решение типовых задач обработки мультимедийных данных;

1.3. Формируемые компетенции, а также перечень планируемых результатов обучения по дисциплине (модулю) (знания, умения владения), сформулированные в компетентностном формате.

Дисциплина (модуль) направлена на формирование следующих компетенций:

- общенаучными (ОК):
  - способен решать проблемы в новой или незнакомой обстановке в междисциплинарном контексте, интегрировать знания, формулировать суждения и выводы в условиях неполной определенности, включая социальные и этические аспекты применения знаний; (ОК-3)
- инструментальными (ИК):
  - способен ставить и решать коммуникативные задачи во всех сферах общения (в том числе межкультурных и междисциплинарных), управлять процессами информационного обмена. Владеет навыками работы с большими массивами информации, способен использовать современные информационно-коммуникационные технологии в конкретной области, включая исследовательский контекст; (ИК-3)
- профессиональными (ПК):
  - способен разрабатывать методические и нормативные документы, техническую документацию, проекты и программы. (ПК-12)
-

В результате освоения дисциплины обучающийся должен демонстрировать следующие результаты образования:

1. Знать:
   - основные различия между аналоговыми и цифровыми сигналами и знать проблемы, возникающие при переходе от одного вида представления к другому.
2. Уметь:
   - ориентироваться в современной литературе, относящейся к области цифровых сигналов и изображений.
3. Владеть:
   - теоретическими знаниями в области фильтрации цифровых сигналов и методами сжатия цифровых сигналов.

1.4. Место дисциплины (модулей) в структуре ООП ВПО

Дисциплина (модуль) «Алгоритмические основы мультимедийных технологий» является частью общенаучного цикла (блока) дисциплин учебного плана по направлению подготовки 710100 «Информатика и вычислительная техника», специальности Компьютерные информационные системы. Для освоения дисциплины (модулей) необходимы компетенции, сформированные в ходе изучения следующих дисциплин и прохождения практик: Основы программирования, Математическая логика и теория алгоритмов.

2. Структура дисциплины (модулей)

Структура дисциплины (модулей) для очной формы обучения

Общая трудоемкость дисциплины составляет 3 кредита, 90 ч., в том числе аудиторная работа обучающихся с преподавателем 48 ч., самостоятельная работа обучающихся 42 ч.

№ п/п	*Раздел, Темы Дисциплины*	*Семестр*	*Виды учебной работы, включая самостоятельную работу студентов и трудоемкость (в часах)*				*Формы текущего контроля успеваемости (по неделям семестра) Форма промежуточной аттестации (по семестрам)*
			лекции	*Сем. Заня /лаб. заня*	*СРС*	*СРСиП*	
	Раздел 1. Современные методы кодирования мультимедийных данных						
1	Основные технологии кодирования мультимедиа информации	1	1	2	1	1	опрос, проверка задания, посещаемость
2	Современные технологии кодирования мультимедиа информации	1	1	2	1	1	опрос, проверка задания, посещаемость
3	Знакомство основными компонентами мультимедийного фреймворка FFmpeg	1	1	2	1	1	опрос, проверка задания, посещаемость
4	Основные компоненты мультимедийного фреймворка FFmpeg	1	1	2	1	1	опрос, проверка задания, посещаемость
5	Специализированные компоненты мультимедийного фреймворка FFmpeg	1	1	2	1	1	Модуль 1
	Раздел 2.						

6	Алгоритм кодирования повторов (RLE)	1	1	2	2	1	опрос, проверка задания, посещаемость
7	Алгоритм преобразования изображений в форматах Netpbm (P1, P2, P3)	1	1	2	2	1	опрос, проверка задания, посещаемость
8	Алгоритм преобразования изображений в форматах Netpbm (P4, P5, P6)	1	1	2	2	1	опрос, проверка задания, посещаемость
9	Мультимедийная библиотека Simple DirectMedia Layer	1	1	2	2	1	
10	Специализированные возможности мультимедийной библиотеки SDML	1	1	2	2	1	Модуль 2
	Раздел 3.						
11	Проектирование медиапроигрывателя	1	1	2	2	1	опрос, проверка задания, посещаемость
12	Алгоритмы воспроизведения видеопотока	1	1	2	2	1	опрос, проверка задания, посещаемость
13	Алгоритмы воспроизведения аудиопотока	1	1	2	2	1	Модуль 3
	Раздел 4.						
14	Алгоритмы синхронизации воспроизведения мультимедиа-данных	1	1	2	2	1	опрос, проверка задания, посещаемость
15	Алгоритмы воспроизведения видеопотока	1	1	2	2	1	опрос, проверка задания, посещаемость
16	Алгоритмы потокового воспроизведения мультимедиа-данных	1	1	2	3	2	Модуль 4

3. Содержание дисциплины (модулей)

№	Наименование раздела, темы дисциплины	Краткое содержание
1	Современные методы кодирования мультимедийных данных	Основные технологии кодирования мультимедиа информации. Современные технологии кодирования мультимедиа информации. Знакомство с основными компонентами мультимедийного фреймворка Ffmpeg. Основные компоненты мультимедийного фреймворка Ffmpeg. Специализированные компоненты мультимедийного фреймворка FFmpeg
2	Кодирования и сжатие информации	Алгоритм кодирования повторов (RLE). Алгоритм преобразования изображений в форматах Netpbm (P1, P2, P3). Алгоритм преобразования изображений в форматах Netpbm (P4, P5, P6).

		Мультимедийная библиотека Simple DirectMedia Layer. Специализированные возможности мультимедийной библиотеки SDML.
3	Прикладные мультимедийные библиотеки	Проектирование медиа проигрывателя. Реализация воспроизведения видеопотока. Реализация воспроизведения аудио потока.
4	Прикладные мультимедийные библиотеки	Алгоритмы синхронизации воспроизведения мультимедиа-данных. Алгоритмы поткового воспроизведения мультимедиа-данных.

4. Конспект лекций

Раздел 1 Современные методы кодирования мультимедийных данных
Тема 1 Основные технологии кодирования мультимедиа информации
Тема 2 Современные технологии кодирования мультимедиа информации
Тема 3 Знакомство с основными компонентами мультимедийного фреймворка FFmpeg
Тема 4 Основные компоненты мультимедийного фреймворка FFmpeg
Тема 5 Специализированные компоненты мультимедийного фреймворка FFmpeg
Раздел 2 Кодирование и сжатие информации
Тема 1 Алгоритм кодирования повторов (RLE)
Тема 2 Алгоритм преобразования изображений в форматах Netpbm (P1, P2, P3)
Тема 3 Алгоритм преобразования изображений в форматах Netpbm (P4, P5, P6)
Тема 4 Мультимедийная библиотека Simple DirectMedia Layer
Тема 5 Специализированные возможности мультимедийной библиотеки SDML
Раздел 3 Прикладные мультимедийные библиотеки
Тема 1 Проектирование медиапроигрывателя
Тема 2 Алгоритмы воспроизведения видеопотока
Тема 3 Алгоритмы воспроизведения аудиопотока
Раздел 4 Прикладные мультимедийные библиотеки
Тема 4 Алгоритмы синхронизации воспроизведения мультимедиа-данных
Тема 5 Алгоритмы потокового воспроизведения мультимедиа-данных

Конспект лекций можно просмотреть в приложении 1.

5. Информационные и образовательные технологии

Информационные и образовательные технологии

№ п/п	Наименование раздела	Виды учебной работы	Формируемые компетенции (указывается код компетенции)	Информационные и образовательные технологии
1	Раздел №1.	Лекция	ОК-3, ИК-3, ПК-12	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ОК-3, ИК-3	Дискуссия, Консультирование с разбором

				абстрактных ситуаций
		Самостоятельная работа	ОК-3, ИК-3, ПК-12	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты
2	Раздел №2.	Лекция	ОК-3, ИК-3, ПК-12	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ОК-3, ИК-3	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ОК-3, ПК-12	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты
3	Раздел №3.	Лекция	ОК-2, ИК-3, ПК-12	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ИК-3, ПК-12	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ОК-3, ИК-3	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты

4	Раздел №4.	Лекция	ОК-2, ИК-3, ПК-12	Лекция-визуализация с применением слайд-проектора, Дискуссия, Лекция с разбором конкретных ситуаций
		Лабораторная работа	ИК-3, ПК-12	Дискуссия, Консультирование с разбором абстрактных ситуаций
		Самостоятельная работа	ОК-3, ИК-3	Использование электронного курса лекций, Консультирование и проверка заданий посредством электронной почты

6. Фонд оценочных средств для текущего, рубежного и итогового контролей по итогам освоению дисциплины (модулей)

6.1. Перечень компетенций с указанием этапов их формирования в процессе освоения дисциплины

№ п/п	Контролируемые разделы дисциплины (модулей)	Код контролируемой компетенции (компетенций)	Наименование оценочного средства
1	Разделы №1, №2, №3, №4	ОК-3	опрос, выполнение лабораторных работ
2	Разделы №1, №2, №3, №4	ОК-3	опрос, выполнение лабораторных работ
3	Разделы №1, №2, №3, №4	ИК-3	опрос, выполнение лабораторных работ
4	Разделы №1, №2, №3, №4	ИК-3	опрос, выполнение лабораторных работ
5	Разделы №1, №2, №3, №4	ПК-12	опрос, выполнение лабораторных работ
6	Разделы №1, №2, №3, №4	ПК-12	опрос, выполнение лабораторных работ
7	Разделы №1, №2, №3, №4	ИК-3	опрос, выполнение лабораторных работ
8	Разделы №1, №2, №3, №4	ОК-3	опрос, выполнение лабораторных работ

9	Разделы №1, №2, №3, №4	ПК-12	опрос, выполнение лабораторных работ

6.2. Методические материалы, определяющие процедуры оценивания знаний, умений, навыков и (или) опыта деятельности

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	1, 2, 3, 4, 5, 6, 7, 8 недели	8 баллов	До 40 баллов
- выполнение лабораторных работ	1, 2, 3, 4, 5, 6, 7, 8 недели	6 баллов	До 30 баллов
- посещаемость	1, 2, 3, 4, 5, 6, 7, 8 недели	0,2	10 баллов
Рубежный контроль: (сдача модуля)	8 неделя	100%×0,2=20 баллов	
Итого за I модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	9, 10, 11, 12, 13, 14, 15 недели	8 баллов	До 40 баллов
- выполнение лабораторных работ	9, 10, 11, 12, 13, 14, 15 недели	6 баллов	До 30 баллов
- посещаемость	9, 10, 11, 12, 13, 14, 15 недели	0,2	10 баллов
Рубежный контроль: (сдача модуля)	15 неделя	100%×0,2=20 баллов	
Итого за II модуль			До 100 баллов
Итоговый контроль (экзамен)	Сессия	ИК = Бср × 0,8 + Бэкз × 0,2	

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос	1, 2, 3, 4, 5, 6, 7, 8 недели	8 баллов	До 40 баллов
- выполнение лабораторных работ	1, 2, 3, 4, 5, 6, 7, 8 недели	6 баллов	До 30 баллов
- посещаемость	1, 2, 3, 4, 5, 6, 7, 8 недели	0,2	10 баллов
Рубежный контроль: (сдача модуля)	8 неделя	100%×0,2=20 баллов	
Итого за I модуль			До 100 баллов

Форма контроля	Срок отчетности	Макс. количество баллов	
		За одну работу	Всего
Текущий контроль: - опрос - выполнение лабораторных работ - посещаемость	 9, 10, 11, 12, 13, 14, 15 недели 9, 10, 11, 12, 13, 14, 15 недели 9, 10, 11, 12, 13, 14, 15 недели	 8 баллов 6 баллов 0,2	 До 40 баллов До 30 баллов 10 баллов
Рубежный контроль: (сдача модуля)	15 неделя	100%×0,2=20 баллов	
Итого за II модуль			До 100 баллов
Итоговый контроль (экзамен)	Сессия	ИК = Бср × 0,8 + Бэкз × 0,2	

Полученный совокупный результат (максимум 100 баллов) конвертируется в традиционную шкалу:

Рейтинговая оценка (баллов)	**Оценка экзамена**
От 0 до 54	неудовлетворительно
от 55 до 69 включительно	удовлетворительно
от 70 до 84 включительно	хорошо
от 85 до 100	отлично

6.3. Описание показателей и критериев оценивания компетенций на различных этапах их формирования, описание шкал оценивания

Текущий контроль (0 - 80 баллов)

При оценивании посещаемости, опроса и выполнении лабораторных работ учитываются:

- посещаемость (10 баллов)
- степень раскрытия содержания материала (25 баллов);
- изложение материала (грамотность речи, точность использования терминологии и символики, логическая последовательность изложения материала (20 баллов);
- знание теории изученных вопросов, сформированность и устойчивость используемых при ответе умений и навыков (25 баллов).

Рубежный контроль (0 – 20 баллов)

При оценивании контрольной работы учитывается:

- полнота выполненной работы (задание выполнено не полностью и/или допущены две и более ошибки или три и более неточности) – 10 баллов;
- обоснованность содержания и выводов работы (задание выполнено полностью, но обоснование содержания и выводов недостаточны, но рассуждения верны) – 5 баллов;
- работа выполнена полностью, в рассуждениях и обосновании нет пробелов или ошибок, возможна одна неточность – 5 баллов.

Итоговый контроль (экзаменационная сессия) – ИК = Бср × 0,8 + Бэкз × 0,2

При проведении итогового контроля обучающийся должен ответить на 3 вопроса (два вопроса теоретического характера и один вопрос практического характера).

При оценивании ответа на вопрос теоретического характера учитывается:

- теоретическое содержание не освоено, знание материала носит фрагментарный характер, наличие грубых ошибок в ответе (0 баллов);
- теоретическое содержание освоено частично, допущено не более двух-трех недочетов (10 баллов);
- теоретическое содержание освоено почти полностью, допущено не более одного-двух недочетов, но обучающийся смог бы их исправить самостоятельно (20 баллов);
- теоретическое содержание освоено полностью, ответ построен по собственному плану (30 баллов).

При оценивании ответа на вопрос практического характера учитывается:

- ответ содержит менее 20% правильного решения (0-9 баллов);
- ответ содержит 21-89 % правильного решения (10-39 баллов);
- ответ содержит 90% и более правильного решения (40 баллов).

6.4. Типовые контрольные задания или иные материалы, необходимые для оценки знаний, умений, навыков и (или) опыта деятельности..

Перечень вопросов:

- Основные технологии кодирования мультимедиа информации
- Современные технологии кодирования мультимедиа информации
- Основные компонентами мультимедийного фреймворка FFmpeg
- Специализированные компоненты мультимедийного фреймворка FFmpeg
- Алгоритм кодирования повторов (RLE)
- Алгоритм преобразования изображений в форматах Netpbm (P1, P2, P3)
- Алгоритм преобразования изображений в форматах Netpbm (P4, P5, P6)
- Особенности мультимедийной библиотеки Simple DirectMedia Layer
- Специализированные возможности мультимедийной библиотеки SDML
- Особенности медиапроигрывателя
- Алгоритмы воспроизведения видеопотока
- Алгоритмы воспроизведения аудиопотока
- Алгоритмы синхронизации воспроизведения мультимедиа-данных
- Алгоритмы поткового воспроизведения мультимедиа-данных

7. Учебно-методическое и информационное обеспечение дисциплины

7.1. Список источников и литературы

- Литература:
  - Основная:
    1. Столов Е.Л. Электронный образовательный ресурс "Цифровая обработка сигналов и изображений" [Электронный ресурс] - . - Режим доступа: http://zilant.kpfu.ru/course/view.php?id=43
    2. Столов Е.Л. Электронный образовательный ресурс "Алгоритмические основы медиа технологий" [Электронный ресурс] - . - Режим доступа: http://zilant.kpfu.ru/course/view.php?id=17362
    3. Столов Е.Л., Нигматуллин Р.Р. Электронный образовательный ресурс "Компьютерное зрение" [Электронный ресурс] - режим доступа: http://zilant.kpfu.ru/course/view.php?id=17266
  - Дополнительная:
    4. Современные технологии и технические средства информатизации[Электронный ресурс]: Учебник / О.В. Шишов. - М.: НИЦ Инфра-М, 2012. - 462 с. . - Режим доступа: http://www.znanium.com/bookread.php?book=263337

5. Программа дисциплины "Алгоритмические основы мультимедийных технологий"; 010300.68 Фундаментальная информатика и информационные технологии
6. Технические средства информатизации [Электронный ресурс]: Учебник / Н.В. Максимов, Т.Л. Партыка, И.И. Попов. - 4-е изд., перераб. и доп. - М.: Форум: НИЦ ИНФРА-М, 2013. - 608 с. - Режим доступа: http://www.znanium.com/bookread.php?book=410390
7. FFmpeg Documentation. [Электронный ресурс]. Режим доступа: http://ffmpeg.org/documentation.html
8. Simple DirectMedia Layer homepage. [Электронный ресурс]. Режим доступа: http://wiki.libsdl.org/FrontPage

7.2. Перечень ресурсов информационно-телекоммуникационной сети «Интернет», необходимый для освоения дисциплины (модулей)

1. Википедия - http://ru.wikipedia.org
2. Интернет-портал образовательных ресурсов по ИТ - http://www.intuit.ru
3. Портал математических интернет-ресурсов - http://www.math.ru/
4. Портал математических интернет-ресурсов - http://www.allmath.com/
5. Портал ресурсов по математике, алгоритмике и ИТ - http://algolist.manual.ru/
6. Электронные ресурсы Kyrlibnet http://arch.kyrlibnet.kgz

8. Перечень учебно-методического обеспечения для самостоятельной работы обучающихся.

8.1. Планы практических (семинарских) и лабораторных занятий. Методические указания по организации и проведению

- Тема 1 (2 ч.) Основные технологии кодирования мультимедиа информации
  - Цель занятия: изучение основных технологий кодирования мультимедиа информации
  - Форма проведения – дискуссия
  - Содержание занятия:
    - Реализация возможностей основных технологий кодирования мультимедиа информации
  - Литература: [1, 2, 3, 4, 5, 6, 7, 8]
  - Материально-техническое обеспечение занятия: ЭВМ
- Тема 2 (2 ч.) Современные технологии кодирования мультимедиа информации
  - Цель занятия: изучение современных технологий кодирования мультимедиа информации
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
    - реализация возможностей современных технологий кодирования мультимедиа информации
  - Литература: [1, 2, 3, 4, 5, 6, 7, 8]
  - Материально-техническое обеспечение занятия: ЭВМ
- Тема 3 (2 ч.) Знакомство с основными компонентами мультимедийного фреймворка FFmpeg
  - Цель занятия: изучение основных компонентов мультимедийного фреймворка FFmpeg
  - Форма проведения – разработка программного обеспечения
  - Содержание занятия:
    - Реализация основных компонентов мультимедийного фреймворка FFmpeg
  - Литература: [7]
  - Материально-техническое обеспечение занятия: ЭВМ
- Тема 4 (2 ч.) Основные компоненты мультимедийного фреймворка FFmpeg

    - Цель занятия: изучение основных компонентов мультимедийного фреймворка FFmpeg
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация основных компонентов мультимедийного фреймворка FFmpeg
    - Литература: [7]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 5 (2 ч.) Специализированные компоненты мультимедийного фреймворка FFmpeg
    - Цель занятия: изучение специализированных компонентов мультимедийного фреймворка FFmpeg
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация специализированных компонентов мультимедийного фреймворка FFmpeg
    - Литература: [7]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 6 (2 ч.) Алгоритм кодирования повторов (RLE)
    - Цель занятия: изучение алгоритма кодирования повторов (RLE)
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация алгоритма кодирования повторов (RLE)
    - Литература: [1, 2]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 7 (2 ч.) Алгоритм преобразования изображений в форматах Netpbm (P1, P2, P3)
    - Цель занятия: изучение алгоритма преобразования изображений в форматах Netpbm (P1, P2, P3)
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация алгоритма преобразования изображений в форматах Netpbm (P1, P2, P3)
    - Литература: [1, 2, 3]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 8 (2 ч.) Алгоритм преобразования изображений в форматах Netpbm (P4, P5, P6)
    - Цель занятия: изучение алгоритма преобразования изображений в форматах Netpbm (P4, P5, P6)
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация алгоритма преобразования изображений в форматах Netpbm (P4, P5, P6)
    - Литература: [1, 2, 3]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 9 (2 ч.) Мультимедийная библиотека Simple DirectMedia Layer
    - Цель занятия: изучение мультимедийной библиотеки Simple DirectMedia Layer
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация основных мультимедийной библиотеки Simple DirectMedia Layer
    - Литература: [8]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 10 (2 ч.) Специализированные возможности мультимедийной библиотеки SDML
    - Цель занятия: изучение специализированных возможностей мультимедийной библиотеки SDML
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:

        - Реализация специализированных возможностей мультимедийной библиотеки SDML
    - Литература: [8]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 11 (2 ч.) Проектирование медиапроигрывателя
    - Цель занятия: проектирование медиапроигрывателя
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация интерфейса медиапроигрывателя
    - Литература: [1, 2, 3]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 12 (2 ч.) Алгоритмы воспроизведения видеопотока
    - Цель занятия: изучение алгоритмов воспроизведения видеопотока
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация воспроизведения видеопотока
    - Литература: [1, 2, 3]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 13 (2 ч.) Алгоритмы воспроизведения аудиопотока
    - Цель занятия: изучение алгоритмов воспроизведения аудиопотока
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация воспроизведения аудиопотока
    - Литература: [1, 2, 3]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 14 (2 ч.) Алгоритмы синхронизации воспроизведения мультимедиа-данных
    - Цель занятия: изучение алгоритмов синхронизации воспроизведения мультимедиа-данных
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реализация алгоритмов синхронизации воспроизведения мультимедиа-данных
    - Литература: [1, 2, 3]
    - Материально-техническое обеспечение занятия: ЭВМ
- Тема 15 (2 ч.) Алгоритмы поткового воспроизведения мультимедиа-данных
    - Цель занятия: изучение алгоритмов поткового воспроизведения мультимедиа-данных
    - Форма проведения – разработка программного обеспечения
    - Содержание занятия:
        - Реалихация алгоритмов поткового воспроизведения мультимедиа-данных
    - Литература: [1, 2, 3]
    - Материально-техническое обеспечение занятия: ЭВМ

8.2. Методические указания для обучающихся по освоению дисциплины (модулей)

Вид работы	Содержание (перечень вопросов)	Трудоемкость самостоятельной работы (в часах)	Рекомендации
Раздел №1.			
Подготовка к лекции №2	Современные технологии кодирования мультимедиа информации	2	[1, 2, 3, 4, 5, 6, 7, 8]

Подготовка к лекции №3	Знакомство с основными компонентами мультимедийного фреймворка FFmpeg	2	[7]
Подготовка к лекции №4	Основные компоненты мультимедийного фреймворка FFmpeg	2	[7]
Подготовка к лекции №5	Специализированные компоненты мультимедийного фреймворка FFmpeg	2	[7]
Подготовка к лекции №6	Алгоритм кодирования повторов (RLE)	2	[1, 2]
Итого		10	
Раздел №2.			
Подготовка к лекции №7	Алгоритм преобразования изображений в форматах Netpbm (P1, P2, P3)	3	[1, 2, 3]
Подготовка к лекции №8	Алгоритм преобразования изображений в форматах Netpbm (P4, P5, P6)	3	[1, 2, 3]
Подготовка к лекции №9	Мультимедийная библиотека Simple DirectMedia Layer	3	[8]
Подготовка к лекции №10	Специализированные возможности мультимедийной библиотеки SDML	3	[8]
Подготовка к лекции №11	Проектирование медиапроигрывателя	3	[1, 2, 3]
Итого		15	
Раздел №3.			
Подготовка к лекции №12	Алгоритмы воспроизведения видеопотока	3	[1, 2, 3]
Подготовка к лекции №13	Алгоритмы воспроизведения аудиопотока	3	[1, 2, 3]
Подготовка к лекции №14	Алгоритмы синхронизации	3	[1, 2, 3]

	воспроизведения мультимедиа-данных		
Подготовка к лекции №15	Алгоритмы поткового воспроизведения мультимедиа-данных	3	[1, 2, 3]
Закрепление пройденного курса		5	[1, 2, 3, 4, 5, 6, 7, 8]
Итого		17	
Итого по дисциплине		42	

8.3. Методические рекомендации по подготовке письменных работ

Разработанное программное обеспечение должно быть предоставлено в скомпилированном виде, а также в виде текстового файла, содержащего исходный код программы.

8.4. Иные материалы

Не предусмотрено

9. Материально-техническое обеспечение дисциплины

Для изучения дисциплины, необходимо следующее оборудование: ЭВМ, проектор.

Требования к аудитории: компьютерный класс, имеющий ЭВМ в количестве идентичном количеству обучающихся, ЭВМ для преподавателя с подключенным проектором, наличие доски и средств для отображения/удаления информации на доске (мел/ветошь, маркер/губка).

10. Приложения

Приложение 1

Указания к заданию 1. Подготовка доклада о технологиях кодирования мультимедиа информации

Изучите и опишите одну из современных технологий, используемых для кодирования и сжатия мультимедийных данных:

Кодирование изображений:

- **BMP** (Bitmap Picture) – формат хранения растровых изображений, разработанный компанией Microsoft;
- **JPEG** (Joint Photographic Experts Group) – формат хранения и метод сжатия цифровых изображений с потерями;
- **GIF** (Graphics Interchange Format) – формат хранения статичных и анимированных растровых изображений на основе кодирования LZW;
- **PNG** (Portable Network Graphics) – свободный формат хранения растровых изображений на основе алгоритма DEFALTE;
- **RAW** – формат цифровой фотографии, содержащий необработанные данные, полученные с фотоматрицы;
- **SVG** (Scalable Vector Graphics) – формат хранения векторных изображений на основе технологии XML;

Кодирование звуковой информации:

    - **MP3** (MPEG-2 Audio Layer III) – один из наиболее популярных форматов цифрового кодирования звуковой информации с потерями;
- **FLAC** (Free Lossless Audio Codec) – свободный формат кодирования звуковой информации и используемый в нем алгоритм сжатия аудио без потерь;
- **Vorbis** – свободный формат кодирования аудиоинформации с потерями на основе MDCT;
- **AAC** (Advanced Audio Coding) – стандартизованный формат кодирования и алгоритм сжатия аудиоданных;

Кодирование видеоданных:

- **H.262** (MPEG-2 Part 2) – стандарт кодирования и сжатия видеоданных;
    - **H.264** (MPEG-4 Part 10) – стандарт цифрового сжатия видео высокого разрешения;
- **Theora** – свободный формат сжатия видеоданных, разработанный в рамках проекта Xiph.Org;
- **VP8** – стандартизованный формат сжатия видеоинформации, разработанной корпорацией Google;

Хранение мультимедийных данных:

- **Matroska Multimedia Container** – открытый формат мультимедийного контейнера для хранения в одном файле множество видео-, аудио-, графической информации и субтитров;
- **MP4** (MPEG-4 Part 14) – формат медиаконтейнера, являющийся частью стандарта MPEG-4;
- **3GP** (Third Generation Partnership Project) – формат мультимедийного контейнера для служб 3G UMTS;
- **SWF** (Small Web Format) – формат контейнера для хранения мультимедийных данных, в том числе векторной графики, анимации и скриптов на языке ActionScript.

Подготовьте презентацию доклада в формате PDF или PowerPoint. Текст доклада впишите в заметки к слайдам. Для подготовки доклада можно использовать официальную документацию рассматриваемых технологий. Примерный план доклада:

1) краткая история создания и развития технологии, ее отличия от других популярных технологий;

2) существующие реализации, поддерживаемые платформы, использование технологии в прикладном ПО;

3) принцип работы и особенности используемого алгоритма кодирования информации;

4) преимущества и недостатки рассмотренной технологии и используемых в ней алгоритмов;

5) литература и web-ресурсы о технологии, использованные при подготовке доклада.

Преобразование графической информации

Указания к заданию 2. Реализация алгоритма кодирования повторов (RLE)

Кодирование длин серий (Run-length encoding, RLE) — простой алгоритм сжатия данных, который оперирует сериями данных, то есть последовательностями, в которых один и тот же символ встречается несколько раз подряд.

При кодировании строка одинаковых символов, составляющих серию, заменяется строкой, которая содержит сам повторяющийся символ и количество его повторов. Алгоритм, таким образом состоит из следующих шагов:

1) подсчитать количество одинаковых символов в цепочке;

2) вместо цепочки, записать символ и количество его повторений. Например:

```
AAABBBBBA > 3A4B1A
```

Данный алгоритм прост и в понимании, и в реализации, но он малоэффективен при применении его в одиночку. Существуют файлы, которые содержат повторяющиеся символы, но не могут быть сжаты описанным выше алгоритмом. Это происходит, когда повторяется не символы, а последовательность символов. В этом случае пользуются расширенным алгоритмом:

1) найти повторяющиеся последовательности символов заданной длины;

2) подсчитать количество повторений данной последовательности;

3) записать последовательности и количество их повторений.

Например, для цепочек длины 2:

```
ABABABBBBBCCCC > 3AB2BB2CC
```

Необходимо разработать консольное приложение на языке C++, получающее на входе последовательность букв русского и английского алфавита в формате ASCII, и возвращающего две последовательности: последовательность символов, закодированную с помощью алгоритма кодирования длин серий, и последовательность символов, полученную из первой путем декодирования.

В соответствии с вашим вариантом, реализуйте алгоритм RLE для:

- повторяющихся символов;
- повторяющихся цепочек длины 2;
- повторяющихся цепочек длины 3;
- повторяющихся цепочек длины 4;
- повторяющихся цепочек длины 5.

1. Создайте консольный проект C++. Реализуйте в функции main() проверку на наличие аргумента (входной последовательности символов).

2. Для работы с последовательностями символов используйте классы:

```
#include <string>
#include <sstream>
...
std::string to_encode = argv[1];
...
std::ostringstream decoded;
decoded << char;
```

3. Для преобразования строковых значений в числовые используйте функцию:

```
#include <stdlib.h>

char buffer[256];
std::string stdBuffer;

i = atoi (buffer);
I = std::stoi(stdBuffer);
```

4. Реализуйте описанный выше алгоритм кодирования в виде функции:

```
std::string encode(const std::string & stingToEncode);
```

Разработайте и реализуйте алгоритм декодирования последовательностей в виде функции:

```
std::string decode(const std::string & stingToDecode);
```

Учтите возможность многозначных коэффициентов повторения цепочек.

5. Реализуйте обработку исключений в случае некорректных входных данных.

Указания к заданию 3. Разработка алгоритма преобразования изображений в форматах Netpbm

Простые форматы хранения изображений Portable Anymap определяют правила для обмена графическими файлами. Эти форматы могут обеспечивать промежуточное представление данных при конвертации растровых графических файлов между разными платформами. Характеристики форматов Netpbm приведены в табл. 1.

Таблица 1

Форматы Netpbm

Код	**Формат**	**Изображение**
P1	Portable bitmap / PBM ASCII	Черно-белое
P2	Portable graymap / PGM ASCII	Оттенки серого
P3	Portable pixmap / PPM ASCII	Цветное
P4	Portable bitmap / PBM Binary	Черно-белое
P5	Portable graymap / PGM Binary	Оттенки серого
P6	Portable pixmap / PPM Binary	Цветное

Файлы в форматах Netpbm состоят из заголовка с метаинформацией и тела с данными о цвете каждого пикселя. Заголовок состоит из следующих элементов:

1) строки с кодом, определяющим формат файла;
2) опциональных строк комментариев, начинающихся с символа #;

3) строки с двумя разделенными пробелом числами, задающих ширину и высоту изображения в пикселях;

4) строки с максимальным значением при кодировании цвета (только для PPM).

Тело файла состоит из списка строк, содержащих разделенные пробелами наборы целых значений, кодирующих цвет каждого пикселя изображения:

- PBM – числа 0 и 1 для черного и белого;
- PGM – числа от 0 до 255 для оттенков серого;
- PPM – тройка чисел от 0 до 255 для кодирования RGB.

Таким образом, пример простейшего файла в формате PBM для кодирования изображения с буквой «H» изображен на рис. 1.

```
P1
# Comment
3 3
1 0 1
1 1 1
1 0 1
```

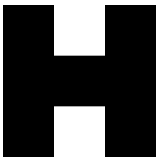

Рис. 1. Пример кодирования изображения в формате PBM ASCII

Для конвертации полноцветных изображений в использующие только оттенки серого существует три основных алгоритма. В методе определения светлоты используется среднее значение между двумя наиболее и наименее значимыми цветами:

```
grayscale = (max(R, G, B) + min(R, G, B)) / 2
```

В методе среднего используется среднее значение всех трёх цветов:

```
grayscale = (R + G + B) / 3
```

В методе определения яркости используется взвешенное среднее значение всех трех цветов, учитывающее человеческое восприятие. Так, поскольку человеческий глаз наиболее восприимчив к зеленому цвету, его вес считается наиболее важным:

```
grayscale = 0.21 * R + 0.71 * G + 0.07 * B
```

Для конвертации изображений в черно-белые необходимо использовать один из существующих алгоритмов дизеринга. Первой и одной из наиболее популярных формул для дизеринга изображений является формула Флойда-Штейнберга. Она основана на переносе ошибки дискретизации значения цвета пикселя на соседние пиксели согласно следующему распределению (где X – обрабатываемый пиксель):

```
        X    7/16
3/16  5/16   1/16
```

Например, если значение обрабатываемого пикселя равно 96 по шкале оттенков серого от 0 (черный) до 255 (белый), а при конвертации его цвет приравнивается к черному, ошибка

дискретизации составит 96. Распределив данную ошибку на соседние пиксели в соответствии с формулой (поделив значение на 16 и умножив на соответствующий коэффициент) получим:

```
     X  +42
+18    +30    +6
```

Необходимо разработать консольное приложение на языке C++, конвертирующее файл из одного формата Netpbm в другой в соответствии с одним из следующих вариантов:

1. PPM ASCII ↔ PPM Binary
2. PPM ASCII ↔ PBM ASCII
3. PPM ASCII ↔ PBM Binary
4. PPM ASCII ↔ PGM ASCII
5. PPM ASCII ↔ PGM Binary
6. PPM Binary ↔ PBM ASCII
7. PPM Binary ↔ PBM Binary
8. PPM Binary ↔ PGM ASCII
9. PPM Binary ↔ PGM Binary
10. PGM ASCII ↔ PGM Binary
11. PGM ASCII ↔ PBM ASCII
12. PGM ASCII ↔ PBM Binary
13. PGM Binary ↔ PBM ASCII
14. PGM Binary ↔ PBM Binary
15. PBM ASCII ↔ PBM Binary

Имя исходного файла передается первым аргументом при запуске приложения. Преобразованное изображение должно иметь такое же название с измененным расширением и/или постфиксом «-converted».

1. Создайте консольный проект C++. Реализуйте в функции main() проверку на наличие аргумента (имени файла).

2. Реализуйте открытие файла для чтения с помощью функций класса fstream.

```
#include <fstream>
#include <string>
int main(int argc, char *argv[])
{
    std::ifstream file("image.pbm");
    std::string str;
    while (std::getline(file, str))
    {
        // Обработка строки
    }
}
```

3. Реализуйте функцию чтения метаданных изображения, хранящихся в заголовке Netpbm-файла.

4. Реализуйте функцию чтения данных о пикселях изображения в двумерный массив.

5. Реализуйте две функции для преобразования изображений:

```
void P1toP4(int pixels[][]);
void P4toP1(int pixels[][]);
```

6. Реализуйте функцию для преобразования изображений:

```
#include <iostream>
std::ofstream outfile ("image.pgm");
outfile << data << std::endl;
```

Прикладные мультимедийные библиотеки

Указания к заданию 4. Знакомство основными компонентами мультимедийного фреймворка FFmpeg

FFmpeg – набор библиотек с открытым исходным кодом, которые позволяют записывать, конвертировать и передавать цифровые аудио- и видеозаписи в различных форматах.

FFmpeg состоит из следующих основных компонентов:

- ffmpeg – утилита командной строки для конвертирования медиафайлов;
- ffplay – простой медиапроигрыватель;
- libavcodec – библиотека с алгоритмами кодирования и декодирования мультимедиа-данных;
- libavformat – библиотека с мультиплексорами и демультиплексорами для различных аудио- и видеоформатов;
- libavutil – вспомогательная библиотека со стандартными общими подпрограммами для различных компонентов ffmpeg;
- libswscale – библиотека для конвертации и масштабирования видео.

Подробную документацию по набору библиотек FFmpeg можно найти в официальной документации [1].

Для разработки приложений на основе FFmpeg необходимо скачать и установить следующие файлы:

- исходные коды библиотек, в том числе файл inttypes.h (URL: http://ffmpeg.zeranoe.com/builds/win32/dev/ и https://code.google.com/p/msinttypes/downloads/list)
- скомпилированные DLL-файлы библиотек (URL: http://ffmpeg.zeranoe.com/builds/win32/shared/)

Обратите внимание на разрядность скачиваемых библиотек – поскольку даже находясь под управлением 64-битной системы Visual Studio по умолчанию собирает проекты под архитектуру x86, рекомендуется скачивать 32-битные версии библиотеки.

1. Скачайте и распакуйте последнюю версию исходных кодов библиотеки FFmpeg в некоторую папку *$FFmpeg*. Также скачайте вспомогательный файл *inttypes.h* и поместите его в папку *$FFmpeg/include/libavutil/*. Отредактируйте

находящийся в той же папке файл *common.h*, скорректировав ссылку на добавленный файл:

```
#include "inttypes.h"
```

2. Откройте среду разработки Visual Studio и создайте пустой проект C++. Откройте свойства созданного проекта (рис. 2). На закладке *VC++ Directories*, добавьте распакованные в пункте 1 папки *$FFmpeg/include* и *$FFmpeg/lib* в списки *Include Directories* и *Library Directories* соответственно.

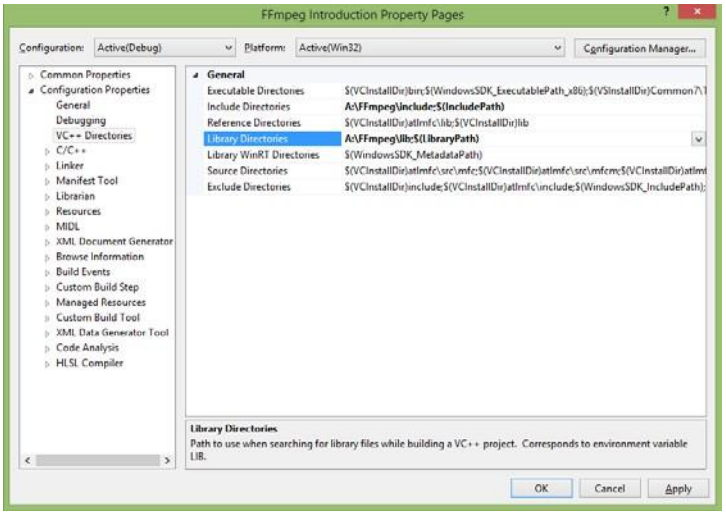

Рис. 2. Окно свойств проекта.

На закладке *Linker/Input* (рис. 3) в список *Additional Dependencies* добавьте библиотеки *avcodec.lib*, *avformat.lib*, *avutil.lib* и *swscale.lib* – этими библиотеками мы будем пользоваться при разработке медиапроигрывателя.

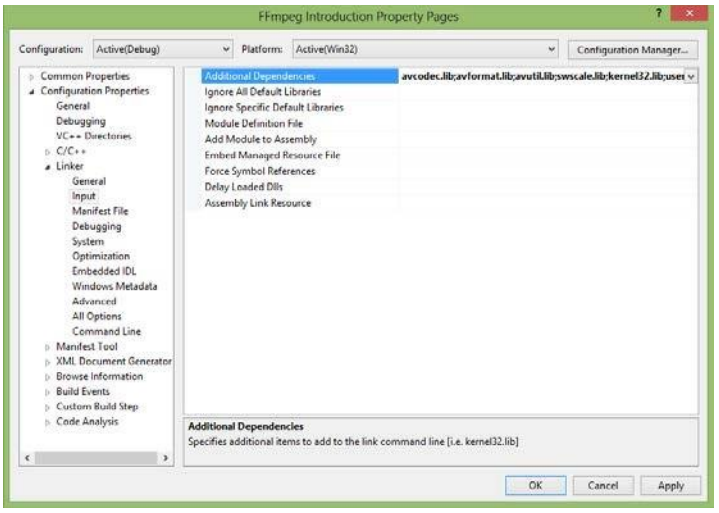

Рис. 3 Окно свойств проекта, закладка Linker.

3. Поскольку библиотека FFmpeg написана на языке C, команды *#include* необходимо обрамить инструкцией *extern "C"*. Создайте новый файл с расширением *.cpp* со следующим содержимым:

```
extern "C"
{
  #include <libavcodec/avcodec.h>
  #include <libavformat/avformat.h>
  #include <libavutil/mem.h>
  #include <libswscale/swscale.h>
}

int main(int argc, char *argv[])
{
  return 0;
}
```

Соберите проект; при возникновении ошибок компоновщика проверьте правильность выполнения предыдущих шагов.

4. Скачайте собранные DLL-файлы библиотеки FFmpeg и распакуйте их в папку с собранной в пункте 3 программой. Для возможности отладки скопируйте их также и в папку с исходными кодами вашей программы. Откройте командную строку и запустите приложение. При возникновении ошибок проверьте правильность размещения файлов библиотек.

5. Перед началом работы необходимо инициализировать библиотеку и зарегистрировать все доступные в ней кодеки с помощью функции:

```
av_register_all();
```

Для открытия файла используйте функцию:

```
AVFormatContext* formatContext = NULL;
avformat_open_input(&formatContext, filename, NULL, NULL);
```

Данная функция читает заголовок файла и сохраняет информацию о медиа- форматах в структуре *AVFormatContext*.

Для вывода основной информации о файле следует воспользоваться функцией:

```
av_dump_format(formatContext, 0, filename, 0);
```

5. Для хранения информации о кодеке, которым закодирован тот или иной медиапоток, используется структура *AVCodecContext*. С ее помощью можно найти и открыть требуемый кодек:

```
AVCodecContext *codecContext = formatContext->streams[i]->codec;
AVCodec *codec = avcodec_find_decoder(codecContext->codec_id);
avcodec_open2(codecContext, codec, NULL);
```

6. Изучите документацию рассмотренных выше функций и классов. Дополните программу проверкой на следующие ошибки: отсутствие аргумента, несуществующий файл, неподдерживаемый формат файла.

Указания к заданию 5. Знакомство с мультимедийной библиотекой Simple DirectMedia Layer

Simple DirectMedia Layer (SDL) — это свободная кроссплатформенная мультимедийная библиотека, реализующая единый программный интерфейс к графической подсистеме, звуковым устройствам и средствам ввода.
Подробную документацию по библиотеке можно найти в официальной документации [2].
Для разработки приложений на основе SDL необходимо скачать и установить следующие файлы:

- исходные коды библиотеки (URL: http://www.libsdl.org/release/SDL2-devel-2.0.0-VC.zip)
- скомпилированные DLL-файлы библиотеки (URL: http://www.libsdl.org/release/SDL2-2.0.0-win32-x86.zip)

1. Скачайте и распакуйте исходные коды библиотеки SDL в некоторую папку *$SDL*.

2. Откройте среду разработки Visual Studio и создайте пустой проект C++. Откройте свойства созданного проекта. Аналогично пункту 2 задания 3, на закладке *VC++ Directories* (рис. 4) добавьте распакованные папки *$SDL/include* и *$SDL/lib/x86* в списки *Include Directories* и *Library Directories* соответственно. На закладке *Linker/Input* в список *Additional Dependencies* добавьте библиотеку *SDL2.lib*.

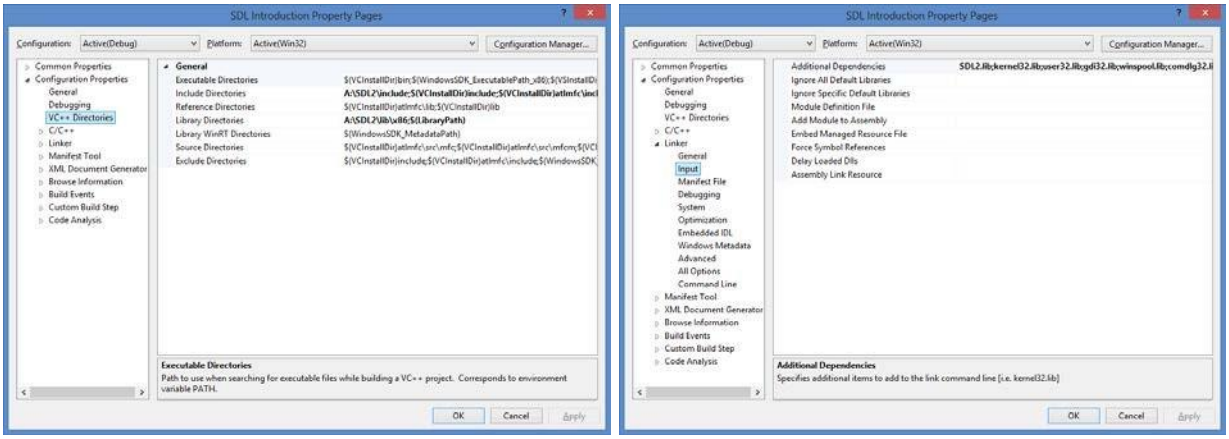

Рис. 4. Окно свойств проекта.

3. Создайте новый файл с расширением *.cpp* со следующим содержимым:

```
#include <SDL.h>
#undef main

int main(int argc, char *argv[])
{
  return 0;
}
```

Поскольку библиотека SDL по умолчанию переопределяет функцию *main()*, для корректной работы требуется отменить данное поведение с помощью инструкции *#undef*. Соберите проект; при возникновении ошибок компоновщика проверьте правильность выполнения предыдущих шагов.

4. Скачайте собранный DLL-файл библиотеки SDL и распакуйте его в папку с собранной в пункте 3 программой. Для возможности отладки скопируйте его также и в папку с исходными кодами вашей программы. Откройте командную строку и запустите приложение. При возникновении ошибок проверьте правильность размещения файлов библиотек.

5. Инициализируйте библиотеку и создайте главное окно приложения:

```
SDL_Init(SDL_INIT_VIDEO | SDL_INIT_AUDIO);
SDL_Window *screen = SDL_CreateWindow("My SDL Window",
                          SDL_WINDOWPOS_UNDEFINED,
                          SDL_WINDOWPOS_UNDEFINED,
                          512, 512,
                          SDL_WINDOW_OPENGL);
```

6. Создайте рендерер:

```
SDL_Renderer *renderer = SDL_CreateRenderer(screen, -1, 0);
```

Для работы с рендерером используются следующие функции:

```
SDL_SetRenderDrawColor(renderer, 0, 0, 0, 255);
SDL_RenderClear(renderer);
SDL_RenderCopy(renderer, texture, NULL, NULL);
SDL_RenderPresent(renderer);
```

7. Для загрузки и отображения текстуры используйте функции:

```
SDL_Surface *surface = SDL_LoadBMP();
SDL_Texture* texture = SDL_CreateTextureFromSurface();
```

Решение типовых задач обработки мультимедийных данных

Указания к заданию 6. Проектирование медиапроигрывателя

Работа с медиа-файлами как правило состоит из следующих блоков:

- чтение;
- демультиплексирование;
- декодирование;
- воспроизведение.

В блоке **чтения** происходит чтение данных из файла или получение потока данных по сети. В библиотеке FFmpeg данные читаются пакетами данных *AVPacket*, которые составлены из одной или нескольких минимальных единиц воспроизведения – фреймов *AVFrame*. Таким образом, чтение файла в FFmpeg производится следующим образом:

```
avformat_open_input (&formatContext, filename, NULL, NULL);
AVPacket packet;
while (av_read_frame(formatContext, &packet) >= 0)
{
  // обработка пакета packet
}
```

В процессе **демультиплексирования** происходит разделение прочитанных из мультимедиа-контейнера данных на соответствующие выходные потоки аудио- и видеоданных. В библиотеке FFmpeg демультиплексирование происходит на уровне пакетов:

```
for (i = 0; i < formatContext->nb_streams; i++)
  if (formatContext->streams[i]->codec->codec_type ==
        AVMEDIA_TYPE_VIDEO)
  {
       videoStreamID = i;
  }


...
if (packet.stream_index == videoStreamID)
{
  // пакет принадлежит видеопотоку
}
```

После демультиплексирования пакетов, происходит раздельное **декодирование** аудио- и видеоданных на уровне фреймов. Сначала производится поиск и запуск необходимого кодека:

```
codecContext = formatContext->streams[videoStreamID]->codec;
codec = avcodec_find_decoder(codecContext->codec_id);
avcodec_open2(codecContext, codec, NULL);
```

После чего происходит декодирование соответствующих пакетов:

```
avcodec_decode_video2(codecContext, &frame, &frameFinished,
&videoPacket);
avcodec_decode_audio4(codecContext, &frame, &frameFinished,
&audioPacket);
```

По окончанию декодирования необходимо **отобразить** полученные кадры видео и **воспроизвести** аудиопоток. Для обоих задач используются промежуточные буферы, которые передаются соответствующим функциям из библиотеки SDL.

Отметим, что воспроизведение аудиоданных в SDL всегда происходит в отдельном порождаемом потоке выполнения, в то время как отображение в SDL (не декодирование!) видеокадров **должно** происходить в главном потоке.

1. На основе описанной схемы работы мультимедиа-проигрывателей, спроектируйте структуру классов медиаплеера на основе библиотек FFmpeg и SDL.

2. Удобнее всего разделить функциональность, связанную с аудио- и видеоданными в различные классы. Опишите, какими методами должны обладать соответствующие компоненты; какие функции библиотек FFmpeg и SDL использовать.

3. Реализуйте прототип основного класса плеера с функциями открытия и проигрывания файлов (за исключением этапа воспроизведения). Не забудьте выделить демультиплексирование и декодирование в отдельные методы разработанных классов! Опишите взаимодействие основного класса проигрывателя со спроектированными вами аудио- и видеокомпонетами.

4. Спроектируйте модель взаимодействия многопоточного выполнения с учетом того, что функции декодирования и воспроизведения аудио- и видеоданных должны происходить параллельно.

Указания к заданию 7. Реализация воспроизведения видеопотока

1. После выполнения задания 6, у вас должен быть реализован прототип медиапроигрывателя, производящего чтение, демультиплексирование и декодирование медиаданных.
Для передачи декодированных видеоданных между библиотеками FFmpeg и SDL следует преобразовывать их в формат **YUV420P**, поддерживаемый обеими библиотеками.

2. На основе функциональности библиотеки *libswscale*, реализуйте метод конвертации видеоизображений в формат YUV420P. Для этого создайте буфер требуемого размера:

```
numPixels = avpicture_get_size (PIX_FMT_YUV420P, width, height);
uint8_t * buffer = malloc (numPixels);
```

3. Далее, инициализируйте контекст преобразования с помощью функции *sws_getCachedContext()*, используя в качестве параметров соответствующие свойства контекста видеокодека. В качестве формата пикселей используйте *PIX_FMT_YUV420P*:

```
struct SwsContext *swsContext = sws_getCachedContext
   ( swsContext,
   codecContext->width,
   codecContext->height,
   codecContext->pix_fmt,
   codecContext->width,
   codecContext->height,
   PIX_FMT_YUV420P,
```

```
    SWS_BILINEAR,
    NULL,
    NULL,
    NULL
);
```

4. С помощью функции *avcodec_alloc_frame()* инициализируйте второй фрейм, в который будет записываться преобразованная копия изображения. Свяжите его с созданным ранее буфером с помощью функции *avpicture_fill()*.

5. Наконец, сконвертируйте изображение в требуемый формат с помощью следующей функции:

```
sws_scale
    ( swsContext,
    frame->data,
    frame->linesize,
    0,
    codecContext->height,
    frameYUV420P->data,
    frameYUV420P->linesize
);
```

3. Создайте пустую текстуру SDL в формате YUV420P, в которую будет помещаться декодированное изображение:

```
SDL_Texture* texture = SDL_CreateTexture
    ( renderer,
    SDL_PIXELFORMAT_IYUV,  // YUV420P
    SDL_TEXTUREACCESS_STREAMING,
    codecContext->width,
    codecContext->height
);
```

4. С помощью функции *SDL_UpdateTexture()* поместите данные из буфера в текстуру. Параметр *pitch* указывает на размер одной строки изображения в байтах; для подсчета этого значения используйте *SDL_BYTESPERPIXEL(),* макрос, возвращающий количество байтов, требуемое для кодирования одного пикселя в заданном формате.

В случае правильной реализации описанного выше алгоритма, видеопоток будет воспроизводиться со скоростью считывания и декодирования пакетов данных, то есть ускоренно. Корректировка скорости воспроизведения будет описана в задании 9.

5. Для очистки памяти и корректного завершения работы библиотек используйте функции:

```
av_free_packet(&packet)
sws_freeContext(swsContext);
av_free(frame);
```

```
avcodec_close(codecContext);
avformat_close_input(formatContext);
SDL_Quit();
```

5. Для обработки внешних событий в библиотеке SDL предусмотрен класс *SDL_Event* и функции *SDL_PollEvent()* и *SDL_WaitEvent()*. С их помощью добавьте возможность выхода из приложения (событие *SDL_QUIT*).

6. Произведите рефакторинг приложения. Определите, насколько реализованная система соответствует спроектированной структуре. Обеспечьте разделение методов инициализации, чтения, демультиплексирования, декодирования и воспроизведения видео.

Указания к заданию 8. Реализация воспроизведения аудиопотока

1. Поскольку воспроизведение аудиоданных в SDL происходит в отдельном потоке выполнения, для обмена данными с главным потоком выполнения (в котором происходит считывание и демультиплексирование данных) необходимо промежуточный буфер пакетов. Реализовать его можно, например, в виде простой очереди:

```
class PacketQueue
{
  int Put(AVPacket *pkt);
  int Get(AVPacket *pkt);
}
```

Для обеспечения целостности данных и синхронизации потоков выполнения используйте методы работы с мьютексами, описанными в библиотеке SDL:

```
mutex = SDL_CreateMutex();
cond = SDL_CreateCond();
...
SDL_LockMutex(mutex);
SDL_UnlockMutex(mutex);
...
SDL_CondWait(cond, mutex);
SDL_CondSignal(cond);
```

Протестируйте разработанный класс. В основном классе проигрывателя создайте экземпляр очереди пакетов для хранения аудиоданных. Реализуйте добавление аудиопакетов в созданную очередь в методе демультиплексирования.

2. Библиотека SDL поддерживает только аудиоданные в формате *S16* (16-bit PCM). Для конвертации аудиоданных из формата *S16P* (16-bit PCM planar) используйте:

```
if (aCodecContext->sample_fmt == AV_SAMPLE_FMT_S16P)
{
```

```
    aCodecContext->request_sample_fmt = AV_SAMPLE_FMT_S16;
}
```

3. Для инициализации аудиокомпонента SDL и подключения к аудиоустройству используйте функцию *SDL_OpenAudio()* со следующими параметрами структуры *SDL_AudioSpec*:

```
SDL_AudioSpec desiredSpecs;
// частота дисктретизации
desiredSpecs.freq = audioCodecContext->sample_rate;
// формат аудиопотока; SDL поддерживает аудио S16
desiredSpecs.format = AUDIO_S16SYS;
// количество каналов
desiredSpecs.channels = aCodecContext->channels;
// размер буфера в сэмплах
desiredSpecs.samples = 1024;
// указатель на функцию обратного вызова
desiredSpecs.callback = callback;
// данные, передаваемые в callback
desiredSpecs.userdata = userdata;
```

Воспроизведение аудиоданных в SDL происходит в отдельном потоке выполнения, который с некоторой заданной частотой вызывает функцию *callback* для наполнения аудиобуфера декодированными данными. Для запуска звука используйте функцию *SDL_PauseAudio()*.

4. Реализуйте функцию обратного вызова со следующей сигнатурой:

```
static void AudioCallback(void *userdata, Uint8 *stream, int streamSize)
```

где *userdata* – указатель на произвольные данные, который вы указали в структуре *SDL_AudioSpec* при подключении к аудиоустройству (данный параметр удобно использовать для получения ссылки на экземпляр медиапроигрывателя или аудиоконтекста); *stream* – указатель на участок памяти, в который помещаются декодированные аудиоданные, готовые к воспроизведению; *streamSize* – максимальный размер этих данных.

В этой функции инициализируйте промежуточный буфер и передайте его в функцию декодирования DecodeAudio (см. следующий пункт данного задания):

```
#define MAX_AUDIO_FRAME_SIZE 192000 // 1 секунда аудио в формате 48khz
32bit
...
static uint8_t audioBuffer[MAX_AUDIO_FRAME_SIZE];
static unsigned int audioBufferSize = 0;
static unsigned int audioBufferIndex = 0;
...
audioBufferSize = DecodeAudio(audioBuffer, sizeof(audioBuffer));
audioBufferIndex = 0;
```

После получения декодированных аудиоданных, скопируйте их в выходной поток stream:

```
   int audioSizeToCopy = audioBufferSize - audioBufferIndex;
   memcpy(stream, (uint8_t *)audioBuffer + audioBufferIndex, audioSizeTo-
Copy);
```

Реализуйте повторение данных инструкций в цикле до полного заполнения потока *stream*.

5. Реализуйте функцию декодирования аудиопотока *DecodeAudio()*, которая будет опрашивать очередь аудиопакетов, декодировать полученные данные с помощью функции *avcodec_decode_audio4()*, и копировать их в выходной буфер *audioBuffer*. Для определения объема памяти, необходимого для хранения декодированного фрейма, воспользуйтесь функцией *av_samples_get_buffer_size()*:

```
int DecodeAudio(uint8_t *audioBuffer)
{
   ...
   dataSize =
      av_samples_get_buffer_size( NULL,
      aCodecContext->channels,
      frame.nb_samples,
      aCodecContext->sample_fmt,
      1
   );
   memcpy(audioBuffer, frame.data[0], dataSize);
}
```

В случае правильной реализации, медиапроигрыватель будет воспроизводить как видео-, так и аудиопотоки. Синхронизация скорости их воспроизведения описана в задании 9.

Указания к заданию 9. Реализация алгоритмов синхронизации воспроизведения мультимедиа-данных

Самым простым способом синхронизации является синхронизация видео к аудио. Для его реализации необходимо рассчитывать текущее значение времени отображения на основе меток **PTS** (Presentation Timestamp) и **DTS** (Decoding Timestamp), характеризующих каждый пакет медиаданных.

Таким образом, алгоритм синхронизации имеет следующий общий вид:

1) вычислить значение времени отображения для текущего видеофрейма;
2) вычислить значение времени воспроизведения текущего аудиосэмпла;
3) вычислить разницу между двумя значениями и на основе полученного значения рассчитать задержку отображения видеокадра.

1. Выделите функцию демультиплексирования медиапотоков и функцию декодирования видеоданных в отдельные потоки исполнения:

```
SDL_Thread *demux = SDL_CreateThread(PlayThread, "demux", this);
SDL_Thread *main = SDL_CreateThread(VideoThread, "video", this);

// Бесконечный цикл с обработкой событий медиаплеера

SDL_WaitThread(demux, NULL);
SDL_WaitThread(main, NULL);
```

Обмен пакетами между созданными потоками организуйте аналогично пункту 1 задания 8. В потоке демультиплексирования:

```
if (packet.stream_index == videoStream)
  videoQueue->Put(&packet);
```

В потоке декодирования видео:

```
while (...)
{
  videoQueue->Get(&videoPacket);
  avcodec_decode_video2(vCodecContext, frame, &frameFinished,
&videoPacket);
  // Преобразование полученного фрейма
  SDL_UpdateTexture(texture, NULL, buffer, pitch);
  ...
}
```

2. Реализуйте отрисовку кадров по таймеру (данная функциональность потребуется для реализации задержки и синхронизации видео). Для создания таймера используйте функцию *SDL_AddTimer()*, отрисовку видеокадра перенесите в функцию обратного вызова *timerCallback*:

```
SDL_INIT(SDL_INIT_TIMER);
SDL_AddTimer(delay, timerCallback, data);
...
static Uint32 timerCallback (Uint32 interval, void *opaque)
{
  ...
  SDL_RenderCopy(renderer, texture, NULL, NULL);
}
```

Обратите внимание, что функция отрисовки, вызываемая в главном потоке выполнения, и функция декодирования, вызываемая в отдельном потоке (см. пункт 1 задания 9), работают с одной и той же переменной *texture*. Для исключения состояния гонки, воспользуйтесь системой мьютексов, описанной в пункте 1 задания 8.

3. В потоке декодирования видео добавьте переменную для хранения текущего значения времени и метод его обновления:

```
double
clock;

double UpdateClock(AVFrame* frame)
{
  if (frame->pkt_dts != AV_NOPTS_VALUE)
       clock = av_q2d(videoStream->time_base) *
  frame->pkt_dts; else if (frame->pkt_pts != AV_NOPTS_VALUE)
       clock = av_q2d(videoStream->time_base) *
  frame->pkt_pts; return clock;
}
```

Реализуйте аналогичный метод для хранения времени воспроизведения аудио, добавив в него вычисление поправки значения времени в случае отсутствия обеих меток **PTS** и **DTS**:

```
clock += (double) dataSize
  /
  ( codecContext->channels
  *
     codecContext->sample_rate *
  av_get_bytes_per_sample(codecContext->sample_
  fmt)
);
```

4. Реализуйте метод вычисления задержки отображения текущего кадра: рассчитав задержку между текущим значением времени отображения кадра с предыдущим использованным, сравните ее с разницей в значениях времени аудио и видео. Определите продолжительность необходимой задержки (или ее отсутствие в случае опережения видеопотоком аудио) до следующего кадра и запустите таймер с найденным значением задержки.

Описанный алгоритм не является идеальным и предназначен для воспроизведения наиболее корректных файлов. В случае правильной реализации, медиапроигрыватель будет способен воспроизводить медиафайлы полностью корректно и синхронизировано.