МУНИЦИПАЛЬНОЕ БЮДЖЕТНОЕ УЧРЕЖДЕНИЕ
ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ
ГОРОДА РОСТОВА-НА-ДОНУ
«ДВОРЕЦ ТВОРЧЕСТВА ДЕТЕЙ И МОЛОДЕЖИ»

ЦЕНТР ЦИФРОВОГО ОБРАЗОВАНИЯ ДЕТЕЙ «IT-куб»

Принято
педагогическим советом МБУ ДО ДТДМ
Протокол №1 от 31.08.2023 г.
Одобрено
методическим советом МБУ ДО ДТДМ
Протокол № 11 от 30.08.2023 г.

Утверждаю
Директор МБУ ДО ДТДМ

_____ Е.Э. Жихарцева

Приказ № 789 от 31.08. 2023 г.

Жихарцева Елена Эдуардовна
Подписан: Жихарцева Елена Эдуардовна
DN: OU=Директор, O=МБУ ДО ДТДМ, CN=Жихарцева Елена Эдуардовна, E=childrentalant@mail.ru
Основание: Я являюсь автором этого документа
Местоположение: место подписания
Дата: 2023-08-31 11:57:13
Foxit Reader Версия: 9.7.2

ДОПОЛНИТЕЛЬНАЯ ОБЩЕОБРАЗОВАТЕЛЬНАЯ ПРОГРАММА

«Мобильная разработка на Kotlin»

Возрастная категория: 14-18 лет.
Срок реализации: 1 год.

Разработчик программы:
Буланов Д.П.,
педагог дополнительного образования
Программу реализует:
Буланов Д.П.,
педагог дополнительного образования.
Методическое сопровождение:
Букатова Е.В., методист.

г. Ростов-на-Дону
2023 г.

ОГЛАВЛЕНИЕ

I. ПОЯСНИТЕЛЬНАЯ ЗАПИСКА

Программа нацелена на развитие знания принципов разработки мобильных приложений, интереса учащихся к практической работе с мобильными устройствами и программами, формирование представлений об основных правилах и методах программирования мобильных устройств, развитие у учащихся логического мышления, конструкторских способностей в процессе моделирования и экспериментов, командной работе над проектом.

Актуальность данной программы обосновано тем, что количество пользователей мобильными телефонами на операционных системах Android, iOS и WindowsPhone растет с каждым днем. Человек с помощью смартфона получает доступ к неограниченной информации: может вести бухгалтерию, планировать мероприятия, развлекаться, просматривая медиаконтент, устанавливать полезные программы и игры. За счет этого рынок мобильных приложений можно смело назвать перспективной сферой, в которой уже работает большое количество людей.

Современный подросток проводит со своим смартфоном основную часть дня. Сегодня специалистами в области информационных технологий разрабатываются мобильные приложения, которые позволяют решать огромное количество задач. Некоторые служат для того, чтобы устанавливать соединение с сетью. Другие помогают оптимизировать маршрут. Третьи предназначены для тех, кто ищет самые выгодные магазины. Есть и такие, с помощью которых можно заказать еду на дом. В связи с этим разработка мобильных приложений является ***актуальным*** и целесообразным в современном мире. Программа «Мобильная разработка на языке Kotlin» научит подростков создавать мобильные разработки, определять значимость и полезность разработки.

Занятия по данной дополнительной образовательной программе смогут помочь ребятам выявить свои интересы и склонности, связанные с разработкой мобильных приложений, программированием. В ходе освоения программы, обучающиеся получают универсальные знания алгоритмов создания программ и применении этих знаний для программирования конкретных приложений под ОС Android.

Отличительная особенность. Дополнительная общеразвивающая программа «Мобильная разработка на языке Kotlin» является модульной. Каждый модуль состоит из кейсов (не менее двух), направленных на формирование определённых компетенций (hard и soft). Результатом каждого кейса является «продукт» (групповой, индивидуальный), демонстрирующий сформированность компетенций. Кейсовые «продукты» могут быть самостоятельным проектом по результатам освоения модуля или общего проекта по результатам всей образовательной программы. Модули и кейсы различаются по сложности и реализуются по принципу «от простого к сложному».

Новизна программы состоит в том, что она учитывает новые технологические уклады, которые требуют новый способ мышления и тесного

взаимодействия при постоянном повышении уровня междисциплинарности проектов, а также использует новые формы диагностики и подведения итогов реализации программы, выполняемые в формате защиты проектов и участия во Всероссийском конкурсе мобильных приложений.

Программа «Мобильная разработка на Kotlin» использует такие методы, как поиск проблем и их практическое решение, анализ и обобщение опыта, подготовка инженерно-технических проектов и их защита, элементы соревнований, неизбежно изменит картину восприятия учащимися технических дисциплин, переводя их из разряда умозрительных в разряд прикладных.

Содержание программы состоит из четырех образовательных модулей и предполагает обучение основам алгоритмизации и программирования на двух уровнях освоения. Первый, второй модуль – стартовый уровень, третий, четвертый модули - базовый уровень освоения.

Направленность. Дополнительная общеобразовательная программа технической направленности (далее – ДОП) «Мобильная разработка на языке Kotlin» ориентирована на учащихся от 14 до 18 лет. Содержание программы рассчитано на 144 академических часа, состоит из четырех образовательных модулей.

Программа обучения состоит из нескольких основных блоков:

- обучение основам языка Kotlin, работе с БД и клиент-серверной архитектурой; базовым принципам ООП;
- знакомство с Android.

Программой предусмотрены следующие виды деятельности обучающихся:

- работа с технической и справочной литературой;
- программирование;
- эксперимент, испытание.

Вид программы – модифицированная. Её содержание разработано в соответствии с требованиями актуальных нормативно-правовых документов в образовании.

При разработке содержания данной программы использованы следующие дополнительные общеобразовательные программы:

- дополнительная общеобразовательная программа технической направленности «Мобильная разработка» /14-18 лет. / Евсеев В.В., Тарасюк В.Л.
- дополнительная общеобразовательная общеразвивающая программа технической направленности «Мобильные приложения» /13-17 лет. / Лямзин М.М.

Уровень освоения – базовый.

Цель программы - формирование технической грамотности средствами приобщения обучающихся к разработке программ под современную платформу Android. Развитие личности и профориентация школьников через повышение их интереса к инженерно-техническим специальностям.

Задачи:

Обучающие:

- расширение знаний о современных и популярных платформах;
- обучение языку программирования Kotlin, языку разметки XML;
- обучение объектно-ориентированному подходу в проектировании и разработке программного обеспечения;
- знакомство с архитектурой приложения под Android;
- обучение программированию технических устройств.

Развивающие:

- формирование алгоритмического мышления;
- формирование навыков работы с информацией;
- формирование умения самостоятельно решать поставленную задачу, излагать мысли в чёткой логической последовательности, отстаивать свою точку зрения, анализировать ситуацию и самостоятельно находить ответы на вопросы путём логических рассуждений;
- развитие логического и технического мышления.

Воспитательные:

- воспитание этики групповой работы и отношений делового сотрудничества;
- создание условий для развития устойчивой потребности в самообразовании.

Прогнозируемые результаты освоения программы обучающимися по уровням

Предметные результаты:

- знание основ языка программирования Kotlin и языка разметки XML;
- понимание принципа работы баз данных и клиент-серверных протоколов;
- умение использовать разные алгоритмы в приёмах программирования;
- умение пользоваться ПК и IDE-разработки для программирования устройства;
- умение читать готовую программу и находить ошибки в готовых программах.

Личностные результаты:

- формирование ответственного отношения к учению, готовности и способности обучающихся к саморазвитию и самообразованию средствами информационных технологий;
- формирование универсальных способов мыслительной деятельности (абстрактно-логического мышления, памяти, внимания, творческого воображения, умения производить логические операции);
- развитие опыта участия в социально значимых проектах, повышение уровня самооценки благодаря реализованным проектам;
- формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками в процессе образовательной, учебно-

исследовательской и проектной деятельности;

- формирование целостного мировоззрения, соответствующего современному уровню развития информационных технологий;
- формирование осознанного позитивного отношения к другому человеку, его мнению, результату его деятельности;
- формирование ценности здорового и безопасного образа жизни; усвоение правил индивидуального и коллективного безопасного поведения при работе с компьютерной техникой.

Познавательные универсальные учебные действия:

- способность к образному и ассоциативному мышлению, фантазии, творческому воображению;
- способность реализовывать на практике основы проектно-исследовательской деятельности;
- умение проводить эксперимент, исследование как под руководством наставника, так и самостоятельно;
- умение вести самостоятельный поиск, анализ, отбор информации, её преобразование, сохранение, передачу и презентацию с использованием ИКТ;
- находить и формировать по результатам наблюдений и исследований зависимости и закономерности.

Коммуникативные универсальные учебные действия:

- умение самостоятельно организовывать целенаправленное учебное взаимодействие в группе;
- способность выражать собственное мнение, отстаивать свою точку зрения, приводить аргументы, подтверждая их фактами;
- умение учитывать разные мнения, сравнивать разные точки зрения;
- умение осуществлять взаимный контроль и оказывать в сотрудничестве необходимую взаимопомощь;
- умение принимать критику к своей работе.

Метапредметные результаты:

- ориентироваться в своей системе знаний: отличать новое знание от известного;
- умение производить анализ поставленной задачи, самостоятельно решать её; формулировать, аргументировать и отстаивать свое мнение; извлекать нужную информацию из открытых источников; составлять примерный алгоритм работы.

Формы учебной деятельности

Учебный процесс строится таким образом, чтобы экспериментальная и практическая работа преобладала над теоретической подготовкой. Необходимые для работы теоретические сведения находятся на каждом персональном компьютере в специальной папке, даются педагогом перед началом практических занятий. Индивидуальная работа проводится во время практических занятий – при выполнении задания у каждого учащегося возникают свои вопросы. Групповая работа проводится во время теоретических

занятий. Каждая тема по программированию сопровождается наглядной демонстрацией работы алгоритма для того, чтобы учащиеся представляли работоспособность алгоритма, а также к чему им нужно стремиться при выполнении поставленной задачи. Учебный процесс организуется на основе постепенного усложнения учебного материала, как теоретического, так и практического.

Программой предусмотрены следующие виды деятельности обучающихся:

- освоение теоретического и практического материала на занятиях;
- разработка индивидуального проекта;
- участие в вебинарах;
- промежуточная аттестация в форме электронного тестирования;
- самостоятельная практическая работа: выполнение домашних заданий, мини-проектов (небольшие приложения, которые реализуются учениками преимущественно на занятиях совместно с учителем с небольшими самостоятельными доработками в качестве домашнего задания).

По типу организации взаимодействия педагогов с обучающимися при реализации программы используются личностно-ориентированные технологии, технологии сотрудничества.

Реализация программы предполагает использование здоровьесберегающих технологий.

Здоровье сберегающая деятельность реализуется:

- через создание безопасных материально-технических условий;
- включением в занятие динамических пауз, периодической смены деятельности обучающихся;
- контролем соблюдения обучающимися правил работы на ПК;
- через создание благоприятного психологического климата в учебной группе в целом.

Объем и срок освоения программы:

Содержание программы рассчитано на 144 часа. Срок освоения программы – 1 год.

Режим занятий: занятия проводятся 2 раза в неделю по 2 урока. Продолжительность занятия - 45 минут. После 45 минут занятий организовывается перерыв длительностью 15 минут для проветривания помещения и отдыха учащихся.

Адресат программы. На обучение по данной программе принимаются обучающие в возрасте от 14 до 18 лет.

Краткое описание возрастных психофизиологических особенностей детей, которым адресовано содержание программы.

Данная программа учитывает возрастные психофизиологические особенности адресата. В возрасте 14-18 лет, молодые люди более обеспокоены формированием собственных убеждений, развитием философского мировоззрения.

В период ранней юности базовыми возрастными потребностями является выбор жизненного маршрута, стремление лучше адаптироваться и соответствовать ожиданиям. Случайно выбранная профессия может негативно сказаться на формировании жизненного пути молодого человека: возникает опыт неудач, занижается самооценка и т.д. В современных социокультурных условиях задача определения жизненного маршрута приобрела особую сложность по ряду причин: ребенок часто не имеет представления о способах и путях достижения конечного результата, а родители и педагоги зачастую сомневаются в правильности своих советов. Перед старшеклассником стоит серьезная задача – определение дальнейшего жизненного маршрута, не просто представлять свое будущее в общих чертах, а осознавать направление движения, задачи и способы достижения поставленных жизненных целей на каждом этапе своего жизненного маршрута. Словом, молодому человеку предстоит в короткие сроки решить вопросы профессионального, личностного и морального самоопределения. В связи с этим наиболее социально приемлема концепция успешности, которая реализуется на занятиях.

Все вышеперечисленные особенности возраста были учтены при разработке содержания программы и технологий её реализации.

Таким образом, настоящая программа является одним из механизмов формирования жизненного маршрута у юношей и девушек, самоопределения в современном обществе, формирует мышление современного человека, основанное на развитии логики с использованием современных компьютерных технологий.

Творческая проектная деятельность обучающихся позволяет наглядно увидеть результаты своей работы и оценить полезность своей работы и значимость развития навыков программирования для жизни. Ребята осваивают основные принципы создания программ с использованием визуальных сред и учатся создавать разнообразные проекты. Учатся презентовать – защищать свой проект перед аудиторией. Важным аспектом является работа обучающихся над индивидуальным проектом.

На основе полученных знаний обучающиеся самостоятельно разрабатывают проект в виде приложения для мобильных платформ на ОС Android. Лучшие проекты могут участвовать в ежегодных школьных хакатонах, проектных сменах Школы IT решений, а также повышают самооценку.

II.УЧЕБНО-ТЕМАТИЧЕСКИЙ ПЛАН

№ п/п	Модуль программы	Количество часов			Форма контроля
		Теория	Практика	Всего часов	
	Стартовая педагогическая диагностика	-	2	2	
1.	**Модуль 1. Основы программирования** (22 часа)				
1.1	Среда разработки	1	1	2	
1.2	Примитивные типы данных. Арифметика	1	1	2	
1.3	Операции отношения и логические операции	1	1	2	
1.4	Условные конструкции. Блоки	1	1	2	
1.5	Циклы. Виды циклов	1	1	2	
1.6	Методы (функции). Видимость переменных	2	2	4	
1.7	Многомерные и неровные массивы	1	3	4	
1.8	Практикум		2	2	
1.9	Контрольное тестирование по модулю		2	2	Тест № 1
2.	**Модуль 2. Объектно-ориентированное программирование** (22 часа)				
2.1	Классы и объекты	1	1	2	
2.2	Классы: конструкторы, статические методы	2	2	4	
2.3	Начальные приёмы тестирования и отладки	1	1	2	
2.4	Архитектура приложений под Android. Активности	1	1	2	
2.5	Интерфейс пользователя. Язык разметки XML	1	1	2	
2.6	Наследование. Намерения	2	2	4	
2.7	Полиморфизм	1	1	2	
2.8	Практикум		2	2	
2.9	Контрольное тестирование по модулю		2	2	Тест № 2
3.	**Модуль 3. Основы программирования Android-приложений** (22 часа)				
3.1	Практикум ООП проектирования	2	2	4	
3.2	Ввод, вывод и исключения	1	1	2	
3.3	Внутренние классы в	2	2	4	

	обработке событий				
3.4	Параллелизм и синхронизация. Потоки	1	1	2	
3.5	Двумерная графика в Android-приложениях	1	1	2	
3.6	Разработка игровых приложений. Реализация графики на основе SurfaceView	1	3	4	
3.7	Практикум		2	2	
3.8	Контрольное тестирование по модулю		2	2	Тест № 3
4.	**Модуль 4. Технологии разработки приложений** (66 часов)				
4.1	Массивы. Алгоритм двоичного поиска	2	2	4	
4.2	Списки	2	2	4	
4.3	Адаптеры в Android	2	4	6	
4.4	Рекурсия	1	3	4	
4.5	Алгоритмы сортировки	2	2	4	
4.6	Ассоциативные массивы	1	3	4	
4.7	Реляционная модель данных.	2	2	4	
4.8	Локальные СУБД. Введение в SQL	2	4	6	
4.9	Практикум		6	6	
4.10	Клиент-серверная архитектура мобильных приложений	2	6	8	
4.11	Облачные платформы. REST-взаимодействие	2	6	8	
4.12	Практикум		6	6	
4.13	Контрольное тестирование по модулю		2	2	Тест № 4
5.0	Работа по индивидуальным проектам		8	8	
6.0	Итоговая защита		2	2	Защита проектов
	Итого:	**43**	**101**	**144**	

III. СОДЕРЖАНИЕ ПРОГРАММЫ

Входная педагогическая диагностика (2 час.)

Практика – 2 часа. Стартовая педагогическая диагностика. Тест на определение начального уровня знаний обучающихся.

Модуль 1. Основы программирования (22 часа)

Среда разработки IntelliJ IDEA/Eclipse. Шаблон программы на Kotlin с функцией main(). О среде разработки IntelliJ IDEA и Eclipse. Понятие проекта. Порядок создания, компиляции, сборки и запуска приложения. Порядок установки среды разработки на домашнем компьютере.

Понятия «бит» и «байт»; двоичная, восьмеричная, шестнадцатеричная системы счисления; перевод чисел из одной системы счисления в другую, понятие переменной.

Переменные и **константы**. Данные, обрабатываемые компьютером; описание переменной; задание начального значения переменной; имя переменной. Понятие «тип переменной». Что относят к примитивным типам. Понятие константы. Ключевое слово final.

Целочисленные типы данных. Разновидности типа «целое»: int, short, long, byte. Размер каждого из типов (количество байт), диапазон представления (минимальное и максимальное значения).

Как задать значение константы в десятичной, двоичной, восьмеричной, шестнадцатеричной системе счисления. Как указать, что константа относится к типу long. Вывод на печать данных целого типа. Ввод данных целого типа.

Типы с плавающей точкой. Типы float, double. Применение суффиксов для указания типа числовых констант.

Арифметические выражения и операторы, операторы присваивания. Сложение, вычитание, умножение, деление, остаток от деления (%), инкремент (++), декремент (--). Префиксная и постфиксная запись инкремента и декремента, уяснение отличия между ними. Операторы присваивания (=, +=, -= и т.д.). Скобки. Рассмотрение подробной таблицы со столбцами: название оператора, форма записи, порядок выполнения. Примеры программ, демонстрирующих применение приоритетов.

Данные типа char. Дать общее представление о кодировке Unicode и UTF-16. Дать рекомендацию по возможности пользоваться не символами, а символьными строками.

Тип данных boolean. Логические значения true и false. Несовместимость типа boolean с int. Отметить, что приведение логических значений к целым и наоборот невозможно.

Логические операции и операции отношения. Операторы отношения: >, <, >=, <=, !=, ==. Уяснение понятия значения операции отношения как ИСТИННО или ЛОЖНО. Логические операции: логическое И, логическое ИЛИ, логическое НЕ. Тернарная операция.

Выражения и **операции**. По итогу изучения различных операций рассмотрение понятия выражения в языке программирования; знаки операций; знаки-разделители. Классификация операций по количеству операндов:

унарные и бинарные. Классификация операций по типу: арифметические, логические, присваивания, отношения и др.

Цикл с предусловием. Синтаксис. Объяснение логики работы, пример использования.

Цикл с постусловием. Синтаксис. Объяснение логики работы, пример использования. Уяснение ключевого отличия от цикла while с предусловием: цикл с постусловием выполняется хотя бы один раз.

Операторы прерывания логики управления программой. Безусловные операторы перехода break, continue.

Массивы. Определение массива как совокупности элементов одного и того же типа, расположенных вплотную друг за другом в памяти. Объявление массива двумя способами. Подчеркнуть необходимость создания массива с помощью new(). Значения, инициализируемые массивом по умолчанию.

Итеративная конструкция. Синтаксис. Логика работы, роль каждой из составных частей. Частные формы записи оператора: Примеры некорректного использования операторов цикла, приводящего к зацикливанию. Вложенные циклы for.

Цикл for each. Синтаксис. Преимущества его применения при работе с массивами в сравнении с обычным for. Отметить, что переменная в цикле for each перебирает не индексы массива, а сами элементы массива.

Определение функции как логически самостоятельной именованной части программы, которой могут передаваться параметры, и которая может возвращать какое-то значение.

Область видимости переменных. Обзорная классификация переменных по области видимости: область класса, область метода, область блока.

Матрицы. Отдельно отметить, что массивы в Kotlin имеют особенность: в языке на самом деле нет многомерных массивов, а только одномерные, т.е. многомерные массивы - это искусственно создаваемые “массивы массивов”. Разбор примеров с матрицами: объявление, инициализация, обращение к элементам. Использование вложенного цикла for each.

Неровные массивы. Проиллюстрировать, как массивы второго уровня могут иметь различную размерность. Как результат – получаем неровные массивы.

Модуль 2. Объектно-ориентированное программирование (22 часа)

Основные понятия. Взгляд на ООП как более естественное по сравнению с процедурным подходом отражение в программировании реалий окружающего мира и отношений между ними. Интуитивно-понятное объяснение понятий: класс, объект (экземпляр класса), переменные (поля) класса, метод класса. Иллюстрация на примерах окружающего мира и примерах школьной математики. Сопоставление каждому понятию некоего утверждения, афористично и емко раскрывающего его суть.

Объект – «всё есть объект», класс – «каждый объект имеет тип», переменные – «каждый объект имеет собственную память, отличную от других объектов», метод – «все объекты определенного типа могут принимать

одинаковые сообщения», программа на ОО-языке – «связка объектов, говорящих друг другу что делать, посылаю сообщения или, иными словами, вызывая методы».

Описание протокола класса. Ключевое слово class как начало описания нового типа данных.

Описание полей класса. Метод класса, его аргументы и возвращаемое значение. Описание метода в протоколе класса.

Создание объекта класса с помощью оператора new. Обращение к полям класса через.

Вызов метода через переменную – объект собственного класса. Интерпретация метода как посылки сообщения объекту.

Инкапсуляция, **наследование**, **полиморфизм**. Общее понятие парадигм ООП инкапсуляция, полиморфизм и наследование на примерах из жизни.

Более подробно остановиться на инкапсуляции: уяснение целесообразности скрытия внутреннего строения объекта и ограничения доступа к его полям – взаимодействие с объектом организуется только через его методы. Взгляд на инкапсуляцию как на средство защиты целостности данных объекта: объект «Интервал» (левая граница не должна стать больше правой).

Конструкторы и **деструкторы**. Конструкторы и деструкторы в Kotlin и их использование. Разновидности конструкторов: без аргументов; с аргументами. Понятие конструктора по умолчанию. Особенности автоматической генерации конструктора по умолчанию.

Перегрузка методов. Уяснение возможности иметь несколько методов с одним и тем же именем, но разными сигнатурами (наборами аргументов) на примере конструктора класса. Распространение концепции перегрузки на любой метод Kotlin. Пример перегрузки конструктора и обычного метода. Необходимость уникального списка типов аргументов для различения перегруженных методов. Уникальность как совокупность количества аргументов, их типов и порядка следования. По типу возвращаемого значения метод не может быть перегружен.

Ключевое слово this. Уяснение смысла ключевого слова this как ссылки на текущий объект. Примеры типового использования: возврат в return ссылки на текущий объект; различение имени локальной переменной и имени поля класса при их совпадении.

Спецификаторы доступа. Ключевое слово public и интерфейсный доступ к объектам класса. Ключевое слово private для описания закрытых полей и методов класса. Понятие доступа класса. Ключевое слово public по отношению к классу (а не члену класса) и его смысл.

Статические методы класса. Ключевое слово static и статические члены класса. Интерпретация статических методов как методов класса в отличие от остальных методов – методов экземпляра. Обращение к статическим членам класса через имя класса (а не переменную – объект).

Инициализация различных **типов данных**. Инициализация полей класса

в конструкторе. Явная и неявная инициализация примитивных типов и объектных ссылок. Инициализация статических данных. Инициализация массива, примеры.

Отладочный вывод и логирование. Вывод значений переменных на экран для последующего анализа их значений. Понятие логирования и его области применения. Вывод отладочных данных с помощью android.util.Log и просмотр этих логов в (Window | Show View | Other... | Android | LogCat.).

Использование отладчика. Пошаговое выполнение программы и просмотр переменных. В отличие от отладочного вывода не требует изменения программы. Однако не позволяет легко просматривать сложные конструкции (сумма элементов массива, диагональ матрицы).

Breakpoint для перехода на конкретную строчку программы (в том числе с указанием условия остановки).

Использование макроса (функции) assert. Проверка инвариантов в программе с помощью макроса assert. Самопроверка программы при каждом её запуске. Проверка входных параметров функции (на примере sqrt или log). Для включения функций assert необходимо установить параметр -ea в Run Configurations -> Arguments -> VM arguments (по умолчанию assert игнорируются, чтобы не замедлять программу).

Тестирование отдельных функций (модульное тестирование). Реализация отдельной функции, проверяющей правильность работы некоторой части программы. Отличие от assert. Выделение легко тестируемой (например, не читающей из входного потока) части программы. Написание модульного теста с помощью библиотеки JUnit.

Работа со строками. Краткий обзор классов String, StringBuffer, StringBuilder. Обратить внимание на то, что объекты класса String являются неизменяемыми.

Знакомство со средой разработки приложений под Android. О среде разработки Android Studio. Порядок создания, компиляции, сборки и запуска в среде. Порядок установки IDE и эмулятора для разработки приложений под Android на домашнем компьютере.

Общая структурная **схема** приложения под Android. Разбор и комментирование схемы (используется схема из developer.android.com). Жизненный цикл Android-приложения.

Активности (Activity) и их жизненный цикл в Android. Создание Activity. Жизненный цикл Activity. Стеки Activity. Состояния Activity.

Отслеживание изменений состояния Activity. Класс Activity, методы onCreate, onStart, onPause, onStop, onRestart, onResume, onDestroy.

Примеры использования XML. XML-формат, одновременно понятный человеку и компьютеру, используется для записи конфигурации программ, для передачи данных по сети, хранения данных и другого.

Привести аналогию языка разметки с описанием полей класса. Использование XML в программировании Android-приложений.

Структура документа и комментарии. Пролог, корневой элемент,

остальные элементы и их разметка, секция CDATA. Способ записи комментария в XML-документе.

Описание ресурсов Android с помощью XML. Описание, назначение файлов res/layout/fragment_main.xml и res/values/strings. Обоснование, для чего используется описание всех строк в отдельном файле (удобный поиск и редактирование всех строк, локализация продукта). Основные элементы интерфейса и их описание в файле fragment_main.xml: EditText, TextView.

Разметки (Layouts). LinearLayout и его ориентации, TableLayout. Описание Layout в xml-файле. Вложенные Layout.

Представления (Views). Примеры представлений: TextView, EditText и ProgressBar. Описание представлений в конфигурационном файле. Поиск представлений по их идентификатору findViewById (R.id.name).

Наследование. Наследование классов как создание новых классов на основе уже существующих. Отношение «является» между классами. Расширение базового класса новыми полями и методами в классе наследнике, характерными именно для производного класса. Примеры наследования (человек – студент, четырёхугольник – ромб и пр.). Взгляд на наследование классов как на естественную возможность повторного использования кода и независимого расширения библиотек классов. Интерфейс как задание набора методов всех классов, его реализующих (задание шаблона). Более подробное использование будет рассмотрено в теме «Полиморфизм».

Инициализация базового класса. Синтаксическое описание наследования классов и реализации интерфейсов, ключевое слово extends и implements. Подобъект базового класса внутри объекта производного класса и необходимость его инициализации. Вызов конструктора базового класса как часть конструктора производного класса. Гибкое манипулирование вызовами конструкторов с помощью ключевого слова super.

Защищенные члены класса. Ключевое слово protected и пример мотивации его использования (приведён в приложении): открытый член класса для доступа из производных классов и закрытый для доступа извне.

Приведение к **базовому типу**. Уяснение ключевого правила: объект производного класса является объектом базового класса, но не наоборот. Пример использования этого правила в Kotlin: объект неявно приводится к типу своего родителя. Стоит сообщить об этом, более подробно это будет разобрано в теме «полиморфизм».

Ключевое слово final. Применение final к примитивным типам данных: значение переменной не может быть изменено. Применение final к объектам: объектная ссылка не может быть изменена, но содержимое объекта – может. Применение final к аргументу метода. Применение final к методу: метод не может быть переопределён в производном классе. Применение final к классу: у этого класса не может быть наследников.

Контекст. **Намерения** (Intents). Что такое Context и его использование. Явные и неявные намерения.

Создание нескольких Activity (и экранов) в одном приложении.

Регистрация их в AndroidManifest.xml. Создание xml файла с описанием вида экрана в папке res/layout. Метода setContentView для загрузки xml файла из res/layout. Создание Intent для перехода на другой экран (метод startActivity).

Общие сведения. Взгляд на полиморфизм как на отделение интерфейса от реализации. Уяснение того, что по-разному реализованными объектами можно управлять, используя один и тот же интерфейс, независимо от того, как реализованы эти объекты. Понятие позднего связывания и его связь с наследованием. Пример с циклическим вызовом одного и того же метода, например, «Повернуть» в массиве геометрических фигур – для каждой фигуры, в зависимости от того, объектом какого производного класса она является, будет вызвана своя реализация этого метода.

Суть понятия полиморфизма и полиморфного метода: в коде метод вызывается через ссылку на объект базового класса; фактический класс объекта определяется во время выполнения программы – позднее связывание; фактически при выполнении программы вызывается реализация метода именно для того класса, к которому принадлежит объект, а не реализация для базового класса.

Преимущества полиморфизма. Объяснение возможностей независимого расширения иерархии классов, предоставляемых полиморфизмом: разработчику нового класса-наследника не нужно ничего менять в уже написанном коде.

Абстрактные классы и методы. Понятие абстрактного класса и абстрактного метода. Синтаксис и мотивация использования. Улучшение кода примера путём объявления базового класса «Фигура» абстрактным.

Модуль 3. Основы программирования Android приложений (22 часа)

Принципы SOLID. Принцип единственной обязанности (Single Responsibility Principle). Принцип открытости/закрытости (Open-Closed Principle). Принцип подстановки Барбары Лисков (Liskov Substitution Principle, кратко – LSP). Принцип разделения интерфейса (Interface Segregation Principle). Принцип инверсии зависимостей (Dependency Inversion Principle).

Диаграммы UML. UML как язык графического описания для объектного моделирования в области разработки программного обеспечения. Знакомство с инструментальной средой для создания UML-диаграмм. Диаграмма классов как подмножество диаграмм UML: основные элементы и синтаксис.

Обработка исключений как средство создания надёжного, помехоустойчивого кода. Основные понятия: исключительная ситуация; обработчик исключения; выбрасывание исключения с аргументом с помощью throw; передача управления из текущего контекста наверх.

Обработка исключения с помощью конструкции try-catch. Возможность соответствия одному блоку try нескольких блоков catch. Основные методы класса Exception. Стратегии обработки исключений: прерывание и возобновление. Пример целесообразности возобновления.

Класс File и его методы: exists(), renameTo(), getAbsolutePath(), canRead(), canWrite(), getName(), length(), isDirectory(), lastModified(). Примеры

возникновения и обработки исключений, возникающих при выполнении методов класса.

Внутренние (вложенные) **классы**. Понятие внутреннего класса. Отличие от наследования. Назначение. Доступ к состоянию объекта с помощью внутреннего класса.

Локальные и **анонимные** внутренние классы. Сущность, синтаксис, назначение.

Обработка событий пользовательского интерфейса. Краткий обзор классов и интерфейсов для обработки событий. Классы Listeners. Использование анонимных классов для реализации обработчиков событий.

Общие понятия. Введение в естественный параллелизм алгоритмов. Оценка улучшения производительности при параллельном выполнении программы. Пример нарушения целостности данных при неаккуратной реализации параллелизма и необходимость синхронизации параллельных ветвей.

Потоки (threads) как средство реализации параллелизма в рамках одного процесса (программы). Общее описание того, как работает многопоточная программа. Некоторые «подводные камни» реализации многопоточности; возможная неопределенность при определении порядка доступа к общему ресурсу.

Процессы и **потоки** в Android. Реализация и запуск AsyncTask. Альтернативные способы создания потокового класса. Наследование от класса Thread и реализация интерфейса Runnable. Предпочтения использования того или иного способа.

Реализация логики потока. Метод run () как реализация логики потока. Запуск потока на выполнение с помощью метода start () и смысл его аргумента как ссылки на объект класса, чей метод run () должен выполнять данный поток. Уяснение фундаментального факта, что вызов start () является асинхронным.

Синхронизация потоков. Объявление метода с помощью ключевого слова syncronized. Применение synchronized к отдельным блокам кода внутри run (). Методы взаимодействия потоков: suspend () – resume (), wait () -notify (). Метод join () как команда одному потоку дождаться завершения выполнения другого.

Блокировки. Понятие мёртвой блокировки (deadlock), механизм её возникновения.

Класс Canvas. Обзор методов и полей класса. Создание собственного View с методом onDraw. Вызов setContentView с вновь созданным классом. Класс Paint. Закраска экрана фоновым цветом. Рисование на canvas круга, прямоугольника. Отрисовка текста. Поворот полотна canvas.rotate и convas.resotre. Отрисовка текста под наклоном.

Двумерная анимация. Обзор методов и полей класса. Создание собственного View с методом onDraw. Вызов setContentView с вновь созданным классом. Класс Paint. Закраска экрана фоновым цветом. Рисование на canvas круга, прямоугольника. Отрисовка текста. Поворот полотна canvas.rotate и

canvas.restore. Отрисовка текста под наклоном. Создание массива абсцисс, ординат и скоростей для движения различных элементов.

Общие подходы для реализации игровых приложений. Последовательные этапы проектирования и реализации игрового приложения. Профессии в мире индустрии игр. Понятие игрового движка и его использование при разработке игры.

Класс SurfaceView. Обзор. Отличие View от SurfaceView. Особенность класса SurfaceView – предоставляет отдельную область для рисования, действия с которой должны быть вынесены в отдельный вспомогательный поток приложения. Методы getHolder(), lockCanvas(), unlockCanvasAndPost().

Упражнение 3.6.1. Разбор примера простейшей игры с анимацией.

Практикум

Практическое занятие по темам модуля. Конкретное содержание определяется учителем.

Контрольное тестирование по модулю

Электронный тест охватывает все темы модуля и в большей части направлен на оценку практических знаний и навыков учеников, полученных в ходе изучения модуля.

Модуль 4. Технологии реализации приложений (66 часов)

Библиотечный класс Arrays. Класс Java.utils.Arrays как набор статических методов, полезных для работы с массивами. Заполнение массива с помощью метода Arrays.fill(). Копирование массива с помощью метода System.arraycopy(). Сравнение массивов на равенство с помощью метода Arrays.equals(): уяснение того факта, что программа должна знать, как же ей сравнивать отдельные элементы массивов на равенство, в связи с чем необходимо иметь реализованным метод equals() для класса, к которому принадлежат элементы массива, если они не относятся к одному из примитивных типов.

Библиотечный класс ArrayList как реализация массива без ограничений на количество элементов. Создание списка, метод add(), метод get(), метод size(). Параметризация списка типом объектов при создании: *ArrayList<Person> list = new ArrayList<Person>()*.

Понятие итератора и его использование для обхода контейнера. Итератор как объект. Фундаментальные методы любого итератора: next() – получить следующий объект контейнера; hasNext() – проверить, если ли ещё объекты в контейнере, т.е. не завершен ли обход.

Последовательный и **двоичный** поиск. Последовательный поиск в произвольном линейном массиве: трудоёмкость n. Двоичный поиск в известной игре угадывания числа. Определение его трудоёмкости как двоичный логарифм n. Поиск в отсортированном массиве с помощью метода Arrays.binarySearch().

Понятие Список. Разновидности структур данных, основанных на списках. Список как базовая структура данных. Классификация структур данных, основанных на списках: по дисциплине обслуживания (стеки, очереди), по структуре (односвязные и двусвязные списки).

Очередь как реализация принципа FIFO (первым пришёл – первым вышел). Работа с очередью осуществляется с обоих концов.

Стек как реализация принципа LIFO (последним пришёл – первым вышел). Работа со стеком осуществляется с одного конца. Обратить внимание на следующее различие: работать с очередью записывающий и считывающий процессы, как правило, могут асинхронно, т.е. один процесс пишет, другой видит, что что-то появилось, и забирает что бы то ни было. Считывание же вершины стека может происходить не в любой ситуации, а только если вершина стека находится в определённом состоянии. Если это состояние не достигнуто, продолжается запись в вершину стека.

Двусвязные и **односвязные** списки. Мотивация введения второго указателя: облегчение навигации по списку в обратном направлении, т.к. сдвиг в односвязном списке назад требует больших трудозатрат, в частности, перемещение назад от конца списка требует прохода от начала по всему списку.

Библиотечный класс LinkedList, Queue, Stack. Практическое занятие по библиотечному классу LinkedList, реализующему связные списки.

Класс LinkedList как реализация связного списка. Методы, специфичные для связного списка: addFirst, addLast,getFirst,getLast,removeFirst, removeLast.

Класс Stack и **интерфейс Queue**. Объяснение, что LinkedList – это одна из реализаций интерфейса очередь, привести примеры других реализаций.

Адаптеры. Для чего нужны адаптеры. Готовые адаптеры в Android: SimpleAdapter, ArrayAdapter. Абстрактный класс BaseAdapter.

Применение адаптеров для обработки событий пользовательского интерфейса. Отображение ListView через адаптер. Методы getView(), getListAdapter().

Понятие рекурсивного вызова в программировании. Умение видеть в задаче ситуацию «часть подобна целому» и осмыслить тот факт, что единовременно работает множество экземпляров рекурсивной функции. Формальные и фактические параметры, локальные и глобальные переменные при перемещении по стеку рекурсивных вызовов.

Линейная и **ветвящаяся** рекурсия. Рассмотрение примеров функций с линейной и ветвящейся рекурсией. Иллюстрация «вручную» работы рекурсивной программы (для каждого экземпляра функции подписать передаваемые аргументы и возвращаемые результаты):

- с линейной рекурсией в виде цепочки;
- с ветвящейся рекурсией в виде дерева рекурсивных.

Демонстрация на полученных иллюстрациях необходимости нерекурсивной заглушки.

Значение алгоритмов сортировки. Важность сортировки как самой по себе, так и для повышения быстродействия других алгоритмов: поиска, вставки, слияния.

Сортировка вставкой как наиболее простой алгоритм сортировки. Пошаговая реализация на примере. Программа, реализующая сортировку вставкой для массива целых чисел. Оценка трудоёмкости как n2.

Быстрая сортировка. Идея алгоритма быстрой сортировки (рекурсивное разбиение массива на два). Пошаговая реализация на примере. Структура программы, реализующей быструю сортировку. Оценка трудоёмкости как n*log(n).

Сортировка массивов с помощью метода Arrays.sort(). Уяснение того факта, что способ сравнения для класса, к которому относятся элементы массива, должен быть известен (для двух элементов нужно как-то определить, кто из них больше). Мотивация наследования интерфейса Comparable и реализации метода compareTo(), чтобы массив мог быть отсортирован.

Сортировка и **поиск** в List-контейнерах. Важно уяснить, что методы sort и binarySearch для списков являются статическими методами базового класса Collections, а не класса Arrays.

Метод hashCode() как метод базового класса Object. Метод hashCode() как реализация хэш-фукнции для контейнера, для которого необходимо обеспечить быструю выборку элементов. Необходимость наличия эффективной фукнции hashCode() для реализации класса, представляющего хэш-таблицу.

Понимание факта, что универсального метода для написания метода hashCode() не существует: алгоритм его вычисления должен основываться на семантике объектов класса, для которого этот метод необходимо перегрузить. Перечисление руководящих принципов написания hashCode():

- возвращение одного и того же значения для одного и того же объекта при каждом вызове;
- вычисление хэш-кода должно основываться исключительно на содержимом объекта, но не на его случайной уникальной информации (например, адресе в памяти);
- множество значений, генерируемое функцией hashCode() на множестве объектов контейнера, должно быть распределено как можно равномернее, чтобы каждому значению соответствовало примерно равное количество объектов контейнера.

Семейства контейнеров – Collections и Map. Наличие в библиотеке двух семейств контейнеров – Collections и Map, и подразделение Collection, в свою очередь, на List и Set. Отличие простых контейнеров от ассоциативных. Общая иерархическая схема контейнерных библиотечных классов Kotlin.

Контейнер Set. Set как контейнер уникальных объектов. Наличие двух реализаций: HashSet и TreeSet. Необходимость наличия метода equals() для обоих.

Контейнер HashSet. Уяснение принципиального требования: для класса, объекты которого помещаются в данный контейнер, должен быть написан метод hashCode().

Ассоциативный массив как набор пар «ключ-значение». Требование уникальности ключа. Примеры, когда естественным индексом выступает не целое число, а произвольная символьная строка (например, «государство – столица»). Ознакомление учащихся с фактом, что многомерный ассоциативный массив с произвольным набором индексов, называемый глобал, выступает в

качестве основной структуры данных в нереляционных БД, и для работы с ним существует набор специальных функций.

Общие методы Map: put(), get(), containsKey(), containsValue().

Контейнер HashMap. Контейнер HashMap и пример его использования. Идея контейнера: поиск ключей с помощью хэш-кодов значений.

Контейнер TreeMap. Контейнер TreeMap, хранение данных, упорядоченных по ключу. Метод subMap(), возвращающий часть дерева.

Потоко-безопасность объектов. Напоминание понятия потоко-безопасности объекта. Возможность сделать контейнер потоко-безопасным. Статические методы класса Collections: synchronizedCollection, synchronizedList, synchronizedSet, synchronizedMap.

Важность БД и СУБД. Использование БД во всех направлениях современной деятельности человека на примерах из повседневной жизни. Необходимо объяснить, почему хранение данных в обычных текстовых файлах не может считаться приемлемым решением для эффективной работы с данными и какие задачи в этой связи возлагаются на СУБД. История развития и классификация СУБД.

Характеристика СУБД как соединения программного обеспечения и данных и неотъемлемо включающей следующие составные части:

- набор файлов, содержащих данные – непосредственно физическая база данных;
- спецификация информационного содержимого физической базы данных – схема данных;
- программное обеспечение, поддерживающее доступ и изменение содержимого базы данных – механизм СУБД;
- язык программирования СУБД, используемый для задания схемы и осуществления доступа к базе данных.

Хранение данных в таблицах. Привести примеры хранения данных в таблице. Описание типов связи и отношений: один к одному, один ко многим, многие ко многим. Примеры отношений: муж – жена (в России в каждый момент времени может быть только один супруг), сын – отец (отец всегда один, детей много), брат – сестра (каждый может иметь много братьев и сестер).

Объяснить способы хранения таких данных в таблицах. Способы хранения дополнительных данных в отношениях.

Локальная СУБД SQLite. Знакомство с локальной СУБД SQLite.

Минипроект 4.1. Записная книжка. Создать БД SQLite «Записная книжка» по спроектированной ранее структуре. На её примере разобрать все изучаемые далее инструкции SQL, создание простейшего Android-приложения. Самостоятельно закончить мини-проект.

Язык запросов SQL. О языке запросов SQL. Создание таблиц. Синтаксис запроса CREATE TABLE, используемого для создания таблицы. Команды ALTER TABLE и DROP TABLE, SHOW TABLES.

Для создания базы данных SQLite проще всего воспользоваться утилитой SQLiteStudio.

Вставка, **изменение** и **удаление** данных из таблицы. Краткое разъяснение и синтаксис команды (INSERT INTO table VALUES …).

Обновление таблиц. Синтаксис запроса UPDATE, используемого для изменения записей таблиц. Ключевое слово WHERE и простейшие фильтры.

Выборка данных. Краткое разъяснение назначения и синтаксис команды SELECT. Форма SELECT * для просмотра всей таблицы. Форма SELECT для просмотра только определённых столбцов таблицы. Ключевое слово FROM; имя таблицы, которая является источником информации для запроса. Команда DELETE FROM table WHERE.

Булевы операции в запросах. Более сложные запросы SELECT. Ключевые слова ORDER BY и DISTINCT.

Агрегация. Вариант SELECT для подсчёта количества, суммы, максимума и минимума, среднего и других агрегаций.

Общие понятия. Интернет и протоколы TCP/IP. Адресация устройств в сетях: для чего она нужна. Что может иметь собственный IP-адрес. Кем назначается IP-адрес и как он выглядит в IPv4. Понятие статического и динамического назначения (DHCP). Понятие внешнего и внутреннего IP-адреса. Знакомство с сервисом http://www.nic.ru/whois/ для определения информации о внешних IP-адресах. Исчерпание IPv4-адресов (о новых возможностях IPv6). Маски подсети. Доменные имена (DNS). URL-ссылки. IP-адрес. Стек протоколов TCP/IP. Порты. Маршрутизация в TCP/IP. DNS. Whois-сервис. URL-ссылка. Уяснить разницу между понятиями сайт, сервер, доменное имя.

Команда ping. Синтаксис. Сценарии поведения, когда связь с адресатом есть и когда её нет. Использование для определения IP-адреса домена и для оценки скорости соединения.

Команда tracert. (traceroute в Linux). Синтаксис. Наблюдения за трассами до различных хостов. Желательно до урока подобрать примеры таких адресов.

Команда nslookup. Синтаксис. Перевод ip-адреса из цифрового вида в доменный и обратно.

Команды ipconfig (ifconfig в Linux). Синтаксис. Команда позволяет узнать свой внутренний IP-адрес, адрес маршрутизатора. Уточнить, что у одного компьютера может быть несколько IP-адресов (для каждого сетевого устройства).

Основные понятия. Структура HTTP-запроса: метод, заголовок, тело. Продемонстрировать HTTP через команду telnet, чтобы показать, что HTTP – обычный текстовый протокол, который, как правило, использует порт 80. Понятие сокета.

Web-сервер. Понятие web-сервера, разбор на примере одной из наиболее популярных и распространенных программ, созданных для веб-разработчиков.

Методы GET и POST. Различие GET и POST запросов следуют из их назначения:

- POST – предназначен для передачи данных серверу и влечёт изменение данных на сервере (например, регистрация пользователя);

- GET – также передаёт данные серверу, но эти данные не влекут за собой значительных изменений на сервере (например, передача параметров поиска, активация аккаунта).

Как следствие, у запроса GET тело отсутствует, у запроса POST оно содержит данные.

Примеры запросов GET и POST, передающих на сервер одну и ту же информацию (необходимо подчеркнуть различие, как выглядит эта информация в запросе GET и POST).

Общие понятия. Структура и схема взаимодействия в клиент-серверных мобильных приложениях. Стандартные способы реализации. Синхронные и асинхронные запросы.

Отправка запросов из Android-приложений. Научиться отправлять запрос из Android-приложения. Для получения доступа в интернет необходимо добавить в манифест <*uses-permission android:name = "android.permission.INTERNET"/*>

Классы HttpClient, HttpResponse, StatusLine.

Упражнение 5.3.1. Разобрать программу для Android, которая реализует задание, аналогичное **Упражнению** 5.2.2. Использовать AsyncTask, чтобы не блокировать пользователя.

Формат JSON и **XML**. Сериализация. Формат JSON, синтаксис, применение. Сравнение с XML. Понятие сериализации. Сохранение сериализованного объекта в файл.

Библиотека Retrofit. Показать преимущества библиотеки Retrofit:

- отсутствие необходимости в реализации запросов к API в отдельном потоке;
- простота кода;
- подключение стандартных библиотек (например, Gson) для автоматической конвертации JSON в объекты;
- динамическое построение запросов;
- обработка ошибок.

Облачные платформы. Назначение и возможности облачных платформ: предоставление провайдером хостинга с готовым набором операционных систем, систем управления базами данных, связующему программному обеспечению, средств разработки и тестирования.

Бесплатные облачные платформы как доступный способ разработки готового к применению клиент-серверного приложения. Обзор возможностей на примере https://www.heroku.com/: языки программирования, СУБД, библиотеки.

Стиль взаимодействия REST. Общие принципы организации взаимодействия приложения с сервером посредством протокола HTTP в стиле REST. Важный принцип REST – сервер не запоминает состояние пользователя между запросами, каждый раз необходимо передавать все необходимые параметры. Понятие token.

Основные методы HTTP-запроса, которые реализуют операции:

получение – GET, добавление – POST, модификация – PUT, удаление – DELETE.

OAuth-авторизация через REST API социальных сетей. OAuth-протокол. Объяснение назначения, принципа работы на примере «ВКонтакте». Token – текстовый ключ, который предоставляет ограниченный временный доступ к данным пользователя в социальной сети.

Упражнение 5.4.2. Работа над проектами. Задание: создать клиент-серверное приложение, которое позволяет авторизоваться через OAuth-авторизацию социальной сети «ВКонтакте» и отправлять информацию о пользователе на сервер.

Контрольное тестирование по модулю

Электронный тест охватывает все темы модуля и в большей части направлен на оценку практических знаний и навыков учеников, полученных в ходе изучения модуля.

Завершающий этап (10 часов)

Работа над индивидуальными проектами и их защита.

IV. МЕТОДИЧЕСКОЕ ОБЕСПЕЧЕНИЕ ПРОГРАММЫ

Дополнительная общеобразовательная общеразвивающая программа «Мобильная разработка на языке Konlin» разработана с учетом современных требований к организации образовательного процесса.

Для успешного овладения содержанием образовательной программы сочетаются различные формы, методы и средства обучения. Для развития фантазии и творческих способностей у обучающихся проводятся занятия, на которых они, решая учебные задачи, создают учебные проекты на основании приобретённых знаний и навыков. Большинство учебных занятий проводится в форме практических занятий.

- устные (беседы, объяснение);
- поисковые (изменение программы для приобретения устройством новых свойств);
- демонстрационные (демонстрация возможностей устройства);
- практические (написание программы, проведение мини-соревнований).

В образовательном процессе используются следующие **методы обучения:**

- *Объяснительно-иллюстративный* – предъявление информации различными способами (объяснение, рассказ, беседа, инструктаж, демонстрация, работа с технологическими картами и др.).

- *Проблемный* – постановка проблемы и самостоятельный поиск её решения обучающимися.

- *Репродуктивный* – воспроизводство знаний и способов деятельности (форма: собирание моделей и конструкций по образцу, беседа, упражнения по аналогу).

Формы организации деятельности:

- занятия коллективные, индивидуально-групповые.

- индивидуальная работа детей, предполагающая самостоятельный поиск различных ресурсов для решения задач.

- участие в соревнованиях различного уровня.

Методическое обеспечение

Методические пособия, разрабатываемые преподавателем с учётом конкретных условий. Техническая библиотека объединения, содержащая справочный материал, учебную и техническую литературу. Индивидуальные задания.

Методическое обеспечение учебного процесса включает разработку преподавателем методических пособий, вариантов демонстрационных программ и справочного материала:

- демонстрационные программы;
- инструкции по настройке среды разработки;
- справочные материалы по терминам ПО.

Материально-технические условия реализации

- компьютерный класс, отвечающий требованиям СанПиН для учреждений дополнительного образования;
- кабинет с 12 рабочими местами для обучающихся, рабочим местом преподавателя;
- качественное освещение.

Наименование оборудования	**Назначение/краткое описание функционала оборудования**	**Количество шт.**
Основное оборудование		
Ноутбук	Intel Core i5, 8 Гб ОЗУ, 500 Гб	1
Планшеты	Samsung Galaxy Tab Active 2 8.0 SM-T395 16GB	12
Монитор	27`` AOC i2790PQU	12
Клавиатура	lenovo	12
Мышь	lenovo	12
Наушники полноразмерные	Sennheiser HD 206	12
Графическая станция	P330 Tower 400W, Core i7-8700, 2x8GB RAM, 256GB SSD, 1TB SATA, DVDRW, GeForce GTX1060 6GB, USB Mouse/Keyboard	12
Источник бесперебойного питания	APC Back-UPS 1100VA, CIS BX1100CI-RS	13
Демонстрационное оборудование		
Интерактивная панель	Prestigio MultiBoard 70	1
МФУ	Canon i-SENSYS MF421dw	1
Крепление для интерактивной панели	Prestigio PMBWMK	1
Графическая станция	Lenovo P330 Tower 400W, Core i7-8700, 2x16GB RAM, 256GB SSD, 4TB SATA, DVDRW, GeForce GTX1080 8GB, USB Mouse/Keyboard	1
Вспомогательное оборудование и аксессуары		
Доступ в интернет не менее 10 Мбит/с на класс		

V. ДИАГНОСТИЧЕСКИЙ ИНСТРУМЕНТАРИЙ

Формы контроля результатов освоения программы

- Начальная и итоговая диагностика позволяет выявить начальный уровень подготовки и оценить результативность программы.
- Включенное педагогическое наблюдение помогает на всех этапах освоения программы отслеживать качество усвоения учениками знаний и умений.
- Итоговый контроль по окончании обучения проводится итоговая аттестация в форме публичной защиты проектов.

Классификация занятий по дидактической цели

№	Классификация занятий по дидактической цели	Форма занятия
1.	Изучение и первичное закрепление нового учебного материала	Фронтальная, индивидуальная
2.	Комплексное применение знаний	Индивидуальная, коллективная, групповая
3.	Обобщение и систематизация знаний	Индивидуальная, фронтальная
4.	Актуализация знаний и умений	Индивидуальная, групповая
5.	Контроль и коррекция знаний и умений	Индивидуальная, групповая

Входной контроль - имеет диагностические задачи и осуществляется в начале учебного года. Цель предварительной диагностики – зафиксировать начальный уровень подготовки обучающихся, имеющиеся знания, умения и навыки, связанные с предстоящей деятельностью. Входной контроль может проводиться в следующих формах: творческие работы, самостоятельные работы, вопросники, тестирование и пр.

Промежуточная аттестация проводится на основании диагностики теоретических знаний и практических умений и навыков по итогам освоения модуля. Промежуточная аттестация проводится в следующих формах: защита творческих или исследовательских работ и проектов, конференции, выставочный просмотр, викторины, олимпиада, конкурс, соревнование, турнир и пр.

Итоговая аттестация проводится по окончании обучения по программе.

Возможные уровни теоретической подготовки обучающихся:

Высокий уровень – учащийся освоил практически весь объем знаний (75-100%), предусмотренных программой за конкретный период; специальные термины употребляет осознанно и в полном соответствии с их содержанием.

Средний уровень – у учащегося объем освоенных знаний составляет 50-75%; сочетает специальную терминологию с бытовой.

Низкий уровень – учащийся овладел менее чем 50% объема знаний, предусмотренных программой; учащийся, как правило, избегает употреблять специальные термины.

Возможные уровни практической подготовки обучающихся:

Высокий уровень – учащийся овладел 75-100% умениями и навыками, предусмотренными программой за конкретный период; работает с оборудованием самостоятельно, не испытывает особых трудностей; выполняет практические задания с элементами творчества.

Средний уровень – у учащегося объем усвоенных умений и навыков составляет 50-75%; работает с оборудованием с помощью педагога; в основном выполняет задания на основе образца.

Низкий уровень – учащийся овладел менее чем 50% умений и навыков, предусмотренных программой; испытывает затруднения при работе с оборудованием; обучающийся в состоянии выполнять лишь простейшие практические задания педагога.

Достигнутые обучающимся знания, умения и навыки заносятся в сводную таблицу результатов обучения.

В целях определения уровня усвоения программы учащимися осуществляются диагностические срезы:

- входная диагностика на основе анализа выбранной обучающимися роли в диагностической игре и степени их участия в реализации отдельных ее этапов, где выясняется начальный уровень знаний, умений и навыков учащихся, а так же выявляются их творческие способности;
- промежуточная диагностика позволяет выявить достигнутый на данном этапе уровень знаний, умений и навыков учащихся, в соответствии с реализованной проектной деятельностью. Предлагаются выполнение практических заданий, контрольные тесты;
- итоговая диагностика проводится в конце учебного курса (выставка и защита творческих проектов) и предполагает комплексную проверку образовательных результатов по всем ключевым направлениям. Данный контроль позволяет проанализировать степень усвоения программы учащимися. Результаты контроля фиксируются в диагностической карте.

Диагностика результативности освоения образовательной программы

Оценка освоения обучающимися предметных образовательных результатов программы производится с применением накопительной системы, при которой каждый обучающийся за время обучения по программе может набрать максимально – 100 баллов.

Критерии оценивания:

1) оценивается самостоятельная работа обучающихся, для каждой изучаемой темы определены баллы за освоение теории. Полный балл по теме дается за освоение не менее 80% задач, выделенных на самостоятельное изучение, половина баллов – за 40%.

2) промежуточные практические работы и 1 итоговая. Полный балл по теме дается за решение не менее 100% задач, половина баллов – за решение 50% задач.

Проверка задач по наборам тестов позволяет достаточно точно определить процент выполнения работы. Балл за выполненную работу рассчитывается исходя из процентного соотношения решенных задач к общему количеству задач по разделу.

По результатам аттестации учащийся может получить одну из трёх оценок: незачет (менее 50 баллов), зачет (от 50 до 75 баллов) и зачет с повышенным освоением программы (от 75 до 100 баллов).

При аттестации могут быть учтены достижения учащихся на муниципальном, региональном, федеральном, международном уровнях.

VI. СПИСОК ЛИТЕРАТУРЫ

Нормативная литература

1. Конституция Российской Федерации (принята всенародным голосованием 12.12.1993 с изменениями, одобренными в ходе общероссийского голосования 01.07.2020);
2. Федеральный Закон от 29.12.2012г. № 273-ФЗ «Об образовании в Российской Федерации» (ред.17.02.2023);
3. Федеральный закон РФ от 24.07.1998 № 124-ФЗ «Об основных гарантиях прав ребенка в Российской Федерации» (с изменениями от 29.12.2022);
4. Концепция развития дополнительного образования детей до 2030г., утвержденная распоряжением Правительства Российской Федерации от 31.03. 2022г. №678-р;
5. Стратегия развития воспитания в РФ на период до 2025 года (распоряжение Правительства РФ от 29.05. 2015 г. № 996-р);
6. Государственная программа Российской Федерации «Развитие образования» на 2019-2025 г, утвержденная Постановлением Правительства Российской Федерации от 26 декабря 2017 года № 1642 (ред. от 15.03.2021).
7. Приоритетный проект «Доступное дополнительное образование для детей», утвержденный 30.11.2016г. протоколом заседания президиума при Президенте РФ (в ред.27.09.2017);
8. Федеральный проект "Успех каждого ребенка" (утв. на заседании проектного комитета по национальному проекту "Образование" 07.12.2018 г, пр. 3);
9. Приказ Министерства просвещения РФ от 27.07.2022г. № 629 «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам»;
10. Приказ Министерства просвещения Российской Федерации №467 от 03.09.2019 г. «Об утверждении Целевой модели развития региональных систем дополнительного образования».
11. Распоряжение Министерства просвещения Российской Федерации №Р-126 от 21.06.2021 г. «Об утверждении ведомственной целевой программы «Развитие дополнительного образования детей, выявление и поддержка лиц, проявивших выдающиеся способности».
12. Конвенция о правах ребенка (принята резолюцией 44/25 Генеральной Ассамблеи от 20 ноября 1989 г.) — URL: http://www.un.org/ru/documents/decl_conv/conventions/childcon.shtml.
13. Национальный проект «Образование», утвержденный на заседании президиума Совета при Президенте Российской Федерации по стратегическому развитию и национальным проектам (протокол от 24 декабря 2018 г. № 16).
14. Приказ Министерства просвещения Российской Федерации от 9 ноября 2018 г. № 196 (ред. от 30.09.2020 г.) «Об утверждении Порядка организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам».

15. Приказ Министерства просвещения Российской Федерации от 30 сентября 2020 г. № 533 «О внесении изменений в Порядок организации и осуществления образовательной деятельности по дополнительным общеобразовательным программам, утвержденный приказом Министерства просвещения Российской Федерации от 9 ноября 2018 г. № 196».
16. Приказ Министерства труда и социальной защиты Российской Федерации от 22 сентября 2021 г. N 652н «Об утверждении профессионального стандарта «Педагог дополнительного образования детей и взрослых».
17. Распоряжение Правительства Российской Федерации от 23.01.2021г. № 122-р «Об утверждении Плана основных мероприятий, проводимых в рамках Десятилетия детства, на период до 2027 года.
18. Стратегическая инициатива «Новая модель системы дополнительного образования», одобренная Президентом Российской Федерации 27 мая 2015 г
19. Стратегия развития воспитания в Российской Федерации на период до 2025 года, утвержденная Распоряжением Правительства Российской Федерации от 29 мая 2015 г. № 996-р.
20. Указ Президента Российской Федерации от 29 мая 2017 г. № 240 «Об объявлении в Российской Федерации Десятилетия детства».
21. Указ Президента Российской Федерации от 21 июля 2020 г. № 474 «О национальных целях и стратегических задачах развития Российской Федерации на период до 2030 года».
22. Федеральный закон от 29.12.2012 N 273-ФЗ (ред. от 24.03.2021) «Об образовании в Российской Федерации».
23. Федеральный проект «Успех каждого ребенка», утвержденный президиумом Совета при Президенте Российской Федерации по стратегическому развитию и национальным проектам (протокол от 3 сентября 2018 года № 10).
24. Приказ Министерства образования и науки РФ от 23.08.2017 г. № 816 «Об утверждении Порядка применения организациями, осуществляющими образовательную деятельность, электронного обучения, дистанционных образовательных технологий при реализации образовательных программ»;
25. Приказ Министерства просвещения Российской Федерации от 03.09.2019 № 467 «Об утверждении Целевой модели развития региональных систем дополнительного образования детей» (в ред. от 02.02.2021г.);
26. Постановление Главного государственного санитарного врача РФ от 28.09.2020 N 28 «Об утверждении санитарных правил СП 2.4.3648-20 «Санитарно-эпидемиологические требования к организациям воспитания и обучения, отдыха и оздоровления детей и молодежи»;
27. Постановление Главного государственного санитарного врача РФ от 28.01.2021 № 2 «Об утверждении санитарных правил и норм СанПиН 1.2.3685-21 «Гигиенические нормативы и требования к обеспечению безопасности и (или) безвредности для человека факторов среды обитания» (рзд.VI. Гигиенические нормативы по устройству, содержанию и режиму работы

организаций воспитания и обучения, отдыха и оздоровления детей и молодежи»);
28. Распоряжение правительства Российской Федерации от 29 ноября 2014г. № 2403р. «Основы государственной молодежной политики Российской Федерации на период до 2025 года.»;
29. Государственная программа Ростовской области «Развитие образования», утверждена постановлением Правительства Ростовской области от 17.10.2018 № 646 (с изменениями на 28 декабря 2020 года).
30. Методические рекомендации по проектированию дополнительных общеобразовательных программ (письмо Минобрнауки России от 18 ноября 2015 г. № 09-3242);
31. Областной Закон Ростовской области от 14.11.2013 №26-ЗС «Об образовании в Ростовской области»;
32. Локальные акты МБУ ДО ДТДМ: Устав, Учебный план, Положения о структурных подразделениях, Правила внутреннего трудового распорядка, инструкции по технике безопасности.

Основная литература

1. Зигард Медникс, Лайрд Дорнин, Блейк Мик, Масуми Накамура. Программирование под Android. Programming Android. изд. Питер. 2012 г. 496 стр. ISBN 978-5-459-01115-9, 978-1-449-38969-7.
2. Майер Рето. Android 2. Программирование приложений для планшетных компьютеров и смартфонов. Professional Android 2: Application Developmentecond Edition. изд. Эксмо. 2011 г. 672 стр. ISBN 978-5-699-50323-0.
3. Харди Брайн, Билл Филлипс.Программирование под Android. Android Programming: The Big Nerd Ranch Guide. изд. Питер. 2014 г. 592 стр. ISBN 978-5-496-00502-9, 978-0-321-80433-4.
4. Санитарно-эпидемиологические правила и нормативы СанПиН 2.4.4.3172-14.
5. Распоряжение Правительства РФ от 04.09.2014 № 1726-р «Об утверждении Концепции развития дополнительного образования детей».
6. Портал обучения https://myitschool.ru.
7. Федеральный закон от 29.12.2012 N 273-ФЗ (ред. от 31.12.2014, с изм. от 02.05.2015) «Об образовании в Российской Федерации» (с изм. и доп., вступ. в силу с 31.03.2015).

Литература для обучающихся

1. «Kotlin. Программирование для профессионалов», 2019 год Авторы: Джош Скин, Дэвид Гринхол.
2.Волшебство Kotlin», 2020 год Автор: Пьер-Ив Симон.
3.«Kotlin в действии», 2017 год Авторы: Дмитрий Жемеров, Светлана Исакова.
4.«Head First. Программирование для Android» 2018 год Дэвид Гриффитс, Дон Гриффитс
5.«Android для разработчиков» 3-Е ИЗДАНИЕ 2016 год Пол Дейтел, Харви

Дейтел, Александер Уолд

VII. ПРИЛОЖЕНИЕ

Приложение 1

Календарно-тематический план

№ п/п	Тема занятия	Количество часов			Форма контроля
		Теория	Практика	Всего часов	
1.	Стартовая педагогическая диагностика		2	2	
2.	Среда разработки	1	1	2	
3.	Примитивные типы данных. Арифметика	1	1	2	
4.	Операции отношения и логические операции	1	1	2	
5.	Условные конструкции. Блоки	1	1	2	
6.	Циклы. Виды циклов	1	1	2	
7.	Методы (функции). Видимость переменных	1	1	2	
8.	Методы (функции). Видимость переменных	1	1	2	
9.	Многомерные и неровные массивы	1	1	2	
10.	Многомерные и неровные массивы		2	2	
11.	Практикум		2	2	
12.	Контрольное тестирование по модулю		2	2	Тест № 1
13.	Классы и объекты	1	1	2	
14.	Классы: конструкторы, статические методы	1	1	2	
15.	Классы: конструкторы, статические методы	1	1	2	
16.	Начальные приёмы тестирования и отладки	1	1	2	
17.	Архитектура приложений под Android. Активности	1	1	2	
18.	Интерфейс пользователя. Язык разметки XML	1	1	2	
19.	Наследование. Намерения	1	1	2	
20.	Наследование. Намерения	1	1	2	
21.	Полиморфизм	1	1	2	
22.	Практикум		2	2	
23.	Контрольное тестирование по		2	2	Тест № 2

	модулю				
24.	Практикум ООП проектирования	1	1	2	
25.	Практикум ООП проектирования	1	1	2	
26.	Ввод, вывод и исключения	1	1	2	
27.	Внутренние классы в обработке событий	1	1	2	
28.	Внутренние классы в обработке событий	1	1	2	
29.	Параллелизм и синхронизация. Потоки	1	1	2	
30.	Двумерная графика в Android-приложениях	1	1	2	
31.	Разработка игровых приложений. Реализация графики на основе SurfaceView	1	1	2	
32.	Разработка игровых приложений. Реализация графики на основе SurfaceView		2	2	
33.	Практикум		2	2	
34.	Контрольное тестирование по модулю		2	2	Тест № 3
35.	Массивы. Алгоритм двоичного поиска	1	1	2	
36.	Массивы. Алгоритм двоичного поиска	1	1	2	
37.	Списки	1	1	2	
38.	Списки	1	1	2	
39.	Адаптеры в Android	1	1	2	
40.	Адаптеры в Android	1	1	2	
41.	Адаптеры в Android		2	2	
42.	Рекурсия	1	1	2	
43.	Рекурсия		2	2	
44.	Алгоритмы сортировки	1	1	2	
45.	Алгоритмы сортировки	1	1	2	
46.	Ассоциативные массивы	1	1	2	
47.	Ассоциативные массивы		2	2	
48.	Реляционная модель данных.	1	1	2	
49.	Реляционная модель данных.	1	1	2	

50.	Локальные СУБД. Введение в SQL	1	1	2	
51.	Локальные СУБД. Введение в SQL	1	1	2	
52.	Локальные СУБД. Введение в SQL		2	2	
53.	Практикум		2	2	
54.	Практикум		2	2	
55.	Практикум		2	2	
56.	Клиент-серверная архитектура мобильных	1	1	2	
57.	Клиент-серверная архитектура мобильных	1	1	2	
58.	Клиент-серверная архитектура мобильных		2	2	
59.	Клиент-серверная архитектура мобильных		2	2	
60.	Облачные платформы. REST-взаимодействие	1	1	2	
61.	Облачные платформы. REST-взаимодействие	1	1	2	
62.	Облачные платформы. REST-взаимодействие		2	2	
63.	Облачные платформы. REST-взаимодействие		2	2	
64.	Практикум		2	2	
65.	Практикум		2	2	
66.	Практикум		2	2	
67.	Контрольное тестирование по модулю		2	2	Тест № 4
68.	Работа по индивидуальным проектам		2	2	
69.	Работа по индивидуальным проектам		2	2	
70.	Работа по индивидуальным проектам		2	2	
71.	Работа по индивидуальным проектам		2	2	
72.	Итоговая защита		2	2	Защита проекта
	Итого:	**43**	**101**	**144**	

Приложение 2

Бланк наблюдения за обучающимися

Группа_____

№ п/п	**ФИО**	П О К А З А Т Е Л И					
		Внимателен в течение занятия	Использует базовую систему понятий	Проявляет инициативу, интерес в течение занятия	Идет на деловое сотрудничество	Аккуратно относится к материально-техническим ценностям	Р Е З У Л Ь Т А Т
1							
2							
3							
4							

….

За каждое согласие с утверждением 1 – балл.

Приложение 3

Формы аттестации

Аттестация проводится в форме тестирования после освоения каждого модуля. Оценка производится на основе критериального оценивания.

Итоговая аттестация учащихся осуществляется по 100-балльной шкале, которая переводится в один из уровней освоения образовательной программы согласно таблице:

Набранные баллы учащимся	**Уровень освоения**
0–50 баллов	низкий
50–75 баллов	средний
75–100 баллов	высокий

Распределение баллов и критерии оценивания

№ п/п	Название модуля	Количество баллов	
		минимальное	максимальное
1.	**Модуль 1. Основы программирования**	**5**	**10**
	Посещение занятий	2	3
	Проектная деятельность	3	7
2.	**Модуль 2. Объектно-ориентированное программирование**	**10**	**15**
	Посещение занятий	4	5
	Проектная деятельность	6	10
3.	**Модуль 3. Основы программирования Android-приложений**	**10**	**15**
	Посещение занятий	4	5
	Проектная деятельность	6	10
4.	**Модуль 4. Технологии разработки приложений**	**10**	**20**
	Посещение занятий	4	8
	Проектная деятельность	6	12
5.	**Итоговая защита**	**10**	**40**
	Посещение занятий	4	8
	Проектная деятельность	6	32
Итого:		**45**	**100**

Приложение 4

Входная диагностическая работа

Задание #1

Вопрос:

Алгоритм, в котором его выполнение определяется проверкой каких-либо условий, называется…

Выберите один из 5 вариантов ответа:

1) циклическим
2) следования
3) линейным
4) процедурным
5) разветвляющимся

Задание #2

Вопрос:

Служебное слово IF в условном операторе переводится как...

Выберите один из 5 вариантов ответа:

1) ИНАЧЕ
2) ЕСЛИ
3) ВВОД
4) УСЛОВИЕ
5) ТОГДА

Задание #3

Вопрос:

Служебное слово THEN в условном операторе переводится как...

Выберите один из 5 вариантов ответа:

1) УСЛОВИЕ
2) ВВОД
3) ИНАЧЕ
4) ЕСЛИ
5) ТОГДА

Задание #4

Вопрос:

Служебное слово ELSE в условном операторе переводится как...

Выберите один из 5 вариантов ответа:

1) УСЛОВИЕ
2) ТОГДА
3) ВВОД
4) ИНАЧЕ
5) ЕСЛИ

Задание #5

Вопрос:

Условный оператор
if a mod 2=0 then write (`Да) else write (`Нет`)
позволяет определить, является ли число a:

Выберите один из 4 вариантов ответа:
1) целым
2) двузначным
3) чётным
4) простым

Задание #6

Вопрос:
Статья, набранная на компьютере, содержит 32 страницы, на каждой странице 40 строк, в каждой строке 48 символов. Определите размер статьи в кодировке КОИ-8, в которой каждый символ кодируется 8 битами.
Выберите один из 4 вариантов ответа:
1) 120 Кбайт
2) 480 байт
3) 960 байт
4) 60 Кбайт

Задание #7

Вопрос:
Для какого из приведённых значений числа *X* истинно высказывание: **НЕ**(*X* > 5) **И** (*X* > 4)?
Выберите один из 4 вариантов ответа:
1) 4
2) 5
3) 6
4) 7

Задание #8

Вопрос:
В таблице приведены запросы к поисковому серверу. Для каждого запроса указан его код - соответствующая буква от А до Г. Расположите коды запросов слева направо в порядке возрастания количества страниц, которые нашёл поисковый сервер по каждому запросу. По всем запросам было найдено разное количество страниц. Для обозначения логической операции «ИЛИ» в запросе используется символ «|», а для логической операции «И» - «&»:

Код	**Запрос**
А	Эльфы | Гномы | Орки
Б	Эльфы & Гномы & Орки
В	(Эльфы | Гномы) & Орки
Г	Эльфы | Гномы

Запишите ответ:

_____

Задание #9

Вопрос:
Переведите число 708 из десятичной системы счисления в восьмеричную.
Запишите число:

_____

Задание #10

Вопрос:

Переведите число 319 из десятичной системы счисления в шестнадцатеричную.

Запишите ответ:

_____

Задание #11

Вопрос:

Переведите число 11101010 из двоичной системы счисления в десятичную.

Запишите число:

_____

Задание #12

Вопрос:

Переведите число 541 из восьмеричной системы счисления в десятичную.

Запишите число:

_____

Задание #13

Вопрос:

Переведите число 123 из шестнадцатеричной системы счисления в десятичную.

Запишите число:

_____

Задание #14

Вопрос:

Ниже приведена программа, записанная на трех языках программирования.

C++	Python	Паскаль

#include <iostream>
using namespace std;
 	
int main() {
 int s, k;
 cin >> s;
 cin >> k;
 if (s / 2 == k)
 cout << "ДА";
 else
 cout << "НЕТ";
return 0;
}
	s = int(input())
	k = int(input())
	if s // 2 == k:
	 print("ДА")
	else:
	 print("НЕТ")
		var s, k: integer;
		begin
		 readln(s);
		 readln(k);
		 if s div 2 = k
		 then writeln('ДА')
		 else writeln('НЕТ')
		end.

Было проведено 9 запусков программы, при которых в качестве значений переменных *s* и *k* вводились следующие пары чисел:

(1, 1); (8, 4); (14, 10); (20, 1); (7, 3); (10, 5); (10, 2); (4, 1); (1, 0). Сколько было запусков, при которых программа напечатала «ДА»?

Запишите число: _____

Задание #15

Вопрос:

Среди приведённых ниже трёх чисел, записанных в различных системах счисления, найдите максимальное и запишите его в ответе в десятичной

системе счисления. В ответе запишите только число, основание системы счисления указывать не нужно.
23_{16}, 32_8, 11110_2
Запишите число:

_____

Задание #16

Вопрос:

В языке запросов поискового сервера для обозначения логической операции «ИЛИ» используется символ «|», а для логической операции «И» – символ «&».

В таблице приведены запросы и количество найденных по ним страниц некоторого сегмента сети. Считается, что все запросы выполнялись практически одновременно, так что набор страниц, содержащих все искомые слова, не изменялся за время выполнения запросов.

Запрос	Найдено страниц (в тысячах)
Евгений | Онегин	1700
Евгений	1600
Онегин	1200

Какое количество страниц (в тысячах) будет найдено по запросу *Евгений & Онегин*?

Запишите число:

_____

Задание #17

Вопрос:

В одной из кодировок Unicode каждый символ кодируется 16 битами. Вова написал текст (в нём нет лишних пробелов):

«Чиж, грач, стриж, гагара, пингвин, ласточка, жаворонок, свиристель, буревестник, вертиголовка - птицы».

Ученик вычеркнул из списка название одной птицы. Заодно он вычеркнул ставшие лишними запятые и пробелы - два пробела не должны идти подряд. При этом размер нового предложения в данной кодировке оказался на 12 байт меньше, чем размер исходного предложения. Напишите в ответе вычеркнутое название птицы.

Запишите ответ:

Ответы:

1) (1 б.) Верные ответы: 5;
2) (1 б.) Верные ответы: 2;
3) (1 б.) Верные ответы: 5;
4) (1 б.) Верные ответы: 4;
5) (1 б.) Верные ответы: 3;
6) (1 б.) Верные ответы: 4;
7) (1 б.) Верные ответы: 2;
8) (1 б.) Верный ответ: "БВГА".

9) (1 б.): Верный ответ: 1304.;
10) (1 б.) Верный ответ: "13F".
11) (1 б.): Верный ответ: 234.;
12) (1 б.): Верный ответ: 353.;
13) (1 б.): Верный ответ: 291.;
14) (1 б.): Верный ответ: 4.;
15) (1 б.): Верный ответ: 35.;
16) (1 б.): Верный ответ: 1100.;
17) (1 б.) Верный ответ: "грач".

Интерпретация результатов:
-13-17 баллов – высокий уровень компьютерной грамотности;
-8-12 баллов – средний уровень компьютерной грамотности;
-0-7 баллов – низкий уровень компьютерной грамотности.

Приложение 5

Итоговая диагностическая работа

Итоговая диагностическая работа проводится в конце учебного курса в форме защиты творческого проекта и предполагает комплексную проверку образовательных результатов по всем ключевым направлениям. Данный контроль позволяет проанализировать степень усвоения программы учащимися.

Требуется создать приложение по одной из предложенных тем, также ученик может предложить тему проекта самостоятельно.

- Разработка мессенджера с асинхронным шифрованием данных.
- Разработка мобильной игры движения модели автомобиля по полосе движения и объезда препятствий.
- Написание игры три в ряд с возможностью сетевого взаимодействия и хранения таблицы рекордов.
- Разработка программы для проведения виртуальных экскурсий по дендрарию. Обеспечить возможность прокладки маршрутов и фотографирование.
- Разработка программы «Тетрис» с возможностью сохранения рекордов и сетевого обмена.
- Разработка игры «в города» с фиксацией рекордов.
- Разработка игровой программы для игры в крестики-нолики на бесконечном поле.

Критерии оценивания проектной работы

Представленные учащимся результаты проекта оцениваются в соответствии со следующими критериями:

1) Решение поставленной задачи (максимально 5 баллов)
2) Использование баз данных (максимально 5 баллов)
3) Наличие звукового сопровождения (максимально 5 баллов)
4) Наличие нескольких активностей (экранов) (максимально 5 баллов)
5) Адаптивность приложения (максимально 5 баллов)
6) Клиент-серверное приложение (максимально 5 баллов)
7) Использование принципов MVC (максимально 5 баллов)
8) Ответы на вопросы (максимально 5 баллов)

Максимальное количество баллов – 40

Приложение 6

Диагностика результативности освоения образовательной программы.

Оценка освоения обучающимися предметных образовательных результатов программы производится с применением накопительной системы, при которой каждый обучающийся за время обучения по программе может набрать максимально – 100 баллов.

Критерии оценивания:

1) оценивается самостоятельная работа обучающихся, для каждой изучаемой темы определены баллы за освоение теории. Полный балл по теме дается за освоение не менее 80% задач, выделенных на самостоятельное изучение, половина баллов – за 40%.

2) промежуточные практические работы и 1 итоговая. Полный балл по теме дается за решение не менее 100% задач, половина баллов – за решение 50% задач.

Проверка задач по наборам тестов позволяет достаточно точно определить процент выполнения работы. Балл за выполненную работу рассчитывается исходя из процентного соотношения решенных задач к общему количеству задач по разделу.

По результатам аттестации учащийся может получить одну из трёх оценок: незачет (менее 50 баллов), зачет (от 50 до 75 баллов) и зачет с повышенным освоением программы (от 75 до 100 баллов).

При аттестации могут быть учтены достижения учащихся на муниципальном, региональном, федеральном, международном уровнях.

№ п/п	**Раздел**	**Максимальное количество баллов**
Теория		
1	Основы программирования	3
2	Объектно-ориентированное программирование	5
3	Основы программирования Android-приложений	5
4	Технологии разработки приложений.	8
5	Итоговая защита	8
29 баллов		
Практические работы		
1	Основы программирования	7
2	Объектно-ориентированное программирование	10
3	Основы программирования Android-приложений	10
4	Технологии разработки приложений.	12

5	Итоговая защита	32
71 баллов		
100 баллов		

Диагностика личностных и метапредметных образовательных результатов (в текстовой форме по каждому ожидаемому результату в процентном отношении от общего количества обучающихся группы. Выводы)

Ожидаемый результат	**Параметры**	**Критерии**	**Методы отслеживания**
сформированность способности к саморазвитию	Изобретение школьниками способов решение проблем и развития своих навыков	Наличие зафиксированных попыток	Анализ разрозненной информации
готовность к конструктивному общению и взаимодействию, урегулированию конфликтов в условиях работы в команде при реализации проектов	Соотношение коллективного и индивидуальных результатов	Наличие и адекватность распределения ролей в коллективе в ходе совместного решения проблем. Сравнение коллективного и суммы личных результатов	Наблюдение Беседа Эксперимент
готовность к профессиональному самоопределению	Обращение к педагогу по вопросам смежным с предметом и вопросы о выборе профессии	Количество обращений. Характер вопросов и сообщений, глубина заинтересованности	Статистика (беседы при личной встрече и пр.)
способность к целеполаганию, включая постановку новых для себя целей, преобразование практической задачи в познавательную;	Развитие навыков целеполагания	Адекватность постановки цели, достижимость, результаты	Наблюдение Эксперимент Беседа с родителями

умение планировать пути достижения целей, выбирать средства их реализации и применять данные средства на практике, самоорганизация	Развитие навыков планирования	Количество усвоенных компонент (построение сложных планов, учет взаимосвязей при «распараллеливании работы)	Наблюдение Эксперимент Беседа с родителями
умение оценивать достигнутые результаты, используя критерии оценивания, предложенные наставником или разработанные самостоятельно, самоанализ, самоконтроль и адекватная самооценка.	Наличие умения самостоятельно оценивать результаты своего труда	Степень самостоятельности (участие педагога) Качество усвоения	Самоанализ Беседа Проверка работ