**Осенний конкурс образовательных инноваций 2020**

**Номинация: Оригинальные элементы online-преподавания**

**Автор: Горденко Мария Константиновна**

**Старший преподаватель, младший научный сотрудник, аспирант, Факультет компьютерных наук**

**Общая идея и цель проведения курса с использованием представленной методики**

В данной разработке описаны методы и инструменты проведения лекционных и семинарских занятий в режиме онлайн факультатива «Введение в программирование». Данный курс является адаптационным и предназначен студентов 1 курса бакалавриата образовательной программы «Программная инженерия». В ходе дисциплины студенты знакомятся с языком программирования C#, учатся писать программы, в том числе и в небольших группах, изучают необходимый теоретический материал.

Целями освоения дисциплины «Введение в программирование» являются:

* обеспечение студентов базовыми знаниями по информатике, алгоритмизации и программированию;
* формирования навыков в основах для последующих курсов, связанных с программированием;
* формирование у студентов базовых навыков исследовательской работы, предполагающей самостоятельное изучение специфических инструментов и средств, необходимых для решения именно той конкретной проблемы, которая в качестве задачи поставлена перед ним.

Для реализации курса в онлайн формате используются следующие техники и методики:

* Перевернутый класс;
* Групповые дискуссии;
* Командные упражнения;
* Олимпиада;
* Традиционные формы проведения лекций и семинаров.

Для реализации курса в онлайн формате используется следующее программное обеспечение:

* Kahoot;
* Socrative;
* PeerGrade;
* GitHub Classroom;
* MS Teams;
* Repl.it;
* Telegram;
* NearPod/PearDeck.

**Организация занятий**

В рамках курса предусмотрено проведение как лекционных, так и семинарских занятий. Все занятия проводятся в системе MS Teams[[1]](#footnote-1). Все описанные здесь инструменты рассматриваются подробнее в следующих разделах.

Условно, проводимые в рамках данного курса лекции можно классифицировать как:

* Традиционные лекции;
* Лекции-дискуссии;
* Лекции по модели «перевернутый класс»;
* Видео-лекции.

Рассмотрим каждый тип проводимых лекций подробнее.

**Традиционные лекции или лекции-беседы.** Конечно, при изучении совершенно нового материала не обойтись без традиционных лекций, где преподаватель говорит в течение длительного времени и, возможно, показывает слайды или пишет что-либо на интерактивной доске. Однако, как показывает практика, лекция – это не просмотр интересного остросюжетного фильма, здесь необходимо постоянно быть во внимании, что довольно сложно в рамках онлайна, т.к. преподаватель не видит и не слышит аудиторию, не может проконтролировать то, что делают студенты (действительно ли слушают лекцию). Для того чтобы добавить интерактива в лекцию и проследить, не потерялись ли студенты на пути к знаниям, а заодно и понять степень усваиваемости материала, вместо обычных презентаций, используются презентации NearPod[[2]](#footnote-2) или PearDeck[[3]](#footnote-3). Студенты получают от преподавателя код и по нему присоединяются к обычной презентации, однако, в ходе лекции и прохождения слайдов, получают тестовые вопросы (и не только), на которые необходимо ответить. Рекомендуется делать интерактивные вставки примерно раз в 7-15 минут.

**Лекции-дискуссии или «проблемные» лекции.** Данный тип лекции отличается от предыдущего тем, что не только преподаватель, но и студенты участвуют в лекции, обсуждают некоторые моменты. При подготовке к лекции её необходимо поделить на смысловые блоки, примерно по 15 минут. После каждого смыслового блока студентам задается вопрос, на который не было прямого ответа в лекции, но к нему можно прийти путем рассуждений, опираясь на уже изученный материал. В каких-то случаях, студентам можно предложить найти ответ на вопрос в учебнике или же на какой-то заранее подготовленной веб-страничке. После небольшого обсуждения преподаватель со студентами приходят к какому-либо ответу/выводу и далее продолжается традиционная лекция. Довольно существенным минусом является то, что нельзя опросить всех студентов и, как правило, в процесс вовлекаются только самые активные студенты.

**Лекции по модели «перевернутый класс».** Студенты заранее получают набор теоретического материала для самостоятельного изучения по предстоящей теме. Это может быть учебник/книга, видео-лекция, аудио-лекция и т.д. За отведенное время студенты изучают материал, и на лекции осуществляется уже не прохождение материала заново, а поиск проблемных и трудных мест, уточнение каких-то трудных моментов и вопросов. Вариантами проверки усваиваемости на лекции является устный опрос, интерактивная викторина Kahoot[[4]](#footnote-4) или Socrative[[5]](#footnote-5), решение кейсовых задач по теме студентами в небольших командах по 3-5 человек, дискуссионные игры. Если говорить про устный опрос, то он позволяет понять глубину усвоения материала, но у отдельных студентов. Если студентов на паре много (более 30 человек), то такой опрос удастся провести для малого количества людей и выяснить общее понимание темы будет трудно. В этом плане, для понимания общей картины лучше всего использовать интерактивные викторины Kahoot или Socrative. Они позволяют в режиме реального времени за короткий промежуток времени выяснить проблемные моменты для совместного обсуждения. Что касается кейсовых задач и дискуссионных игр, то они требуют более тщательной подготовки, нежели, чем интерактивные викторины. Однако, обсуждение каких-то вопросов в мини-группах позволяет достичь эффекта самостоятельного поиска решения и обучения не только у преподавателя, но и своих сокурсников, к тому же развивает коммуникативные навыки и сплачивает коллектив студентов. Также такая форма может помочь заменить устный опрос большого числа студентов, т.к. вовлечены в обсуждение будут все студенты, а времени будет затрачено всего 10-20 минут.

**Видео-лекции.** В данном случае преподаватель заранее выбирает видеоматериал по соответствующей теме, после чего его просматривает и делит на логические блоки по 3-7 минут. По каждому логическому блоку готовятся устные вопросы, либо же вопросы в тестирующей системе Socrative. Затем на онлайн-занятии преподаватель совместно со студентами смотрит данное видео (с помощью функции «показать экран»), но с заранее обозначенными логическими остановками через каждые 3-7 минут, чтобы не потерять фокус внимания обучающихся. В ходе каждой остановки происходит обсуждение просмотренного материала. Студенты задают возникающие вопросы преподавателю, а преподаватель, в свою очередь задает заранее заготовленные вопросы по данному блоку студентам. Можно не только задавать вопросы устно, т.к. на них будет отвечать ограниченный круг людей, а подготовить тест в интерактивной тестирующей системе Socrative и в ходе остановок решать по 2-3 тестовых задания на просмотренный блок. Это позволяет преподавателю оценить степень усваиваемости материала, удерживать внимание слушателей и корректировать ход лекции в зависимости от реакции аудитории. Это наименее трудозатратный вариант проведения лекция для преподавателя. Однако, его не стоит применять слишком часто, т.к. такой формат хоть и предполагает присутствие преподавателя и его ответы на вопросы, но все-таки не заменяет настоящей преподавательской лекции; требует тщательного подбора видео-материала (не всегда можно найти необходимый контент); может «не зайти» студентам, находящимся на дистанционном обучении и ежедневно смотрящим десятки обучающих видео.

Если говорить про семинарские занятия, то организация их в онлайн-формате требует определенного рода изобретательности, т.к. необходимо вовлечь студентов в образовательный процесс, а из-за того что они находятся дома, а не в аудитории, то сделать это гораздо сложнее. В рамках данного курса проводятся следующие активности на семинарских занятиях:

* групповые дискуссии;
* обсуждение кейсовых ситуаций в командах;
* командные упражнения;
* соревновательные элементы на дополнительный балл (олимпиады);
* совместное решение тестовых вопросов с последующим обсуждением;
* совместное решение задач с последующим обсуждением;
* самостоятельное решение студентами тестов и задач.

Разберем каждую активность подробнее.

**Групповые дискуссии**. Студенты разбиваются на небольшие команды по 3-5 человек. Получают один или несколько вопросов. В течение 5-15 минут обсуждают вопросы в своей команде, после чего озвучивают ответ перед всей группой и преподавателем. Лучше всего, чтобы у команд были разные вопросы, тогда студенты будут иметь возможность, слушая других, еще и научиться чему-либо или повторить материал.

**Обсуждение кейсовых ситуаций в командах**. Если говорить про программирование, то студентам можно предложить оценить код по заданным критериям в системе Repl.it[[6]](#footnote-6) (чтобы студенты могли его совместно править), предложить альтернативное решение какой-либо программы и т.д. Аналогично групповым дискуссиям, студенты разбиваются на небольшие команды по 3-5 человек, получают кейсовую ситуацию и в течение 5-15 минут обсуждают её решение, а после этого озвучивают свое решение преподавателю.

**Командные упражнения**. В рамках дисциплины «Введение в программирование» студенты получают задачу разработать код для какой-либо задачи в системе GitHub Classroom[[7]](#footnote-7). Данная система позволяет работать в командах и автоматически производить проверку задач, что очень удобно и сокращает время проверки кода на семинарах. Командная работа не только сплачивает ребят, но и позволяет им лучше понять тему засчет объяснения её друг другу.

**Соревновательные элементы на дополнительный балл (олимпиады)**. В ходе курса проводится две большие игры-олимпиады на дополнительные баллы к итоговой оценке. Если студенты занимают в ней первые три места, то получают 1 дополнительный балл. Игры похожи на игру «Математическая карусель», но только проводятся индивидуально. В игре есть два этапа – рубежный и зачетный. Для того чтобы перейти на зачетный этап, студент должен пройти рубежный. Рубежный этап состоит из 20 вопросов в системе Socrative. Для получения зачетных заданий необходимо выполнить тест на 80% и более. Если тест пройти не удалось, то его необходимо начать заново, вопросы при этом меняются и так до тех пор, пока не будет набрано 80%. После того, как студент преодолел рубежный этап, преподаватель выдает ему ссылку на GitHub Classroom с задачами, которые студент должен решить. Цель – решить как можно больше задач на разные темы, но с примерно одинаковой сложностью. Побеждает тот, кто набрал наибольшее количество баллов за задачи. Преимуществом использования системы GitHub Classroom является то, что она имеет возможность проверять код студента по тестам и автоматически выставлять баллы. Таким образом, трудозатраты преподавателя, в основном, направлены на предварительный этап – подготовку вопросов и задач, но на онлайн-семинаре задачей является контроль студентов и ответы на их вопросы.

**Совместное решение тестовых вопросов с последующим обсуждением**. Еще одним видом активности на семинаре является совместное решение тестовых вопросов по пройденной на лекции теме. Система Socrative позволяет настраивать виды взаимодействия с аудиторией. В данном типе активности необходимо выбрать режим, который отображает на экранах всех студентов один и тот же вопрос и переключает его только в случае, если преподаватель дал команду перейти к следующему вопросу. Таким образом, студенты решают тестовый вопрос, преподаватель показывает распределение ответов, после чего происходит обсуждение и переход к следующему вопросу.

**Совместное решение задач с последующим обсуждением**. В данном случае преподаватель дает студентам задание и примерно на 50% решает его вместе со студентами, после чего обсуждается решение оставшейся части задания и студентам дается время на доработку. В рамках данного курса такие задачи также представлены в системе GitHub Classroom, т.к. она позволяет автоматически проверить задачу после сдачи.

**Самостоятельное решение студентами тестов и задач**. Данный вид является традиционным видом семинарских занятий, однако в онлайн формате без должной преподавательской поддержки такие семинары не имеют успеха. Опять же, обязательно должны применяться системы, позволяющие студенту получить мгновенный фидбек (GitHub Classroom для проверки кода, Socrative для проверки тестов). Также преподаватель может в ходе такого семинара задавать вопросы, выявлять проблемные моменты и обсуждать их со студентами.

**Порядок оценивания работы студентов, элементы контроля**

В ходе курса предусмотрено три основных вида элементов контроля и экзамен:

* тест;
* лабораторная работа (со взаимным оцениванием в PeerGrade);
* лабораторная работа (автоматическая оценка в системе GitHub Classroom).

Рассмотрим подробнее каждый элемент контроля.

**Тест** – это набор вопросов с одним или множественным выбором, а также вопросы со словесным ответом. Тесты проводятся в системе Socrative, что позволяет получить автоматически оценку, без ручной проверки преподавателем. В ходе курса предусмотрено 2 теста на оценку (**Т1** и **Т2**).

**Лабораторная работа (со взаимным оцениванием в PeerGrade, Л1 и Л3)** представляет из себя задачу, которая дается студентам на одну неделю. Студенты пишут код и загружают обратно в систему. После чего, студентам дается три дня на проверку работ своих сокурсников. Каждый студент проверяет по пять работ сокурсников и выставляет оценки в соответствии с заранее заданными критериями. Работа студента получает средний балл из всех выставленных отметок. Стоит отметить, что т.к. это ребята 1 года обучения, то процесс оценивания необходимо корректировать, в каких-то ситуациях, работы необходимо пересматривать. Однако, как показывает практика, если заранее обсудить все критерии, то проблем с оценками не возникает. Пример такого задания приведен в приложении.

**Лабораторная работа (автоматическая оценка в системе GitHub Classroom, Л2)** представляет из себя набор из 5-7 задач на одну неделю. Студенты решают данные задачи и загружают в систему GitHub Classroom. До окончания периода сдач студентам доступен ограниченный набор тестовых ситуаций, на которых проверяются программы. После окончания периода сдач задачи проверяются на всех тестах и студенту выставляется балл в соответствии с баллами, набранными на тестах задач.

Экзамен состоит из двух частей – решение теста в Socrative и лабораторная работа из 3 задач с автоматической оценкой в системе GitHub Classroom.

Также предусмотрены неоцениваемые авто проверяемые домашние задания в системе GitHub Classroom. Автоматическая проверка позволяет студентам получить мгновенную обратную связь и самостоятельно найти ошибку либо же задать вопрос преподавателю в чате MS Teams или Telegram-чате, а не ждать, когда преподаватель проверит работу.

По всем видам работ выставляется десятибалльная оценка.

Оценка за лабораторную работу выставляется как средняя результатов взаимного оценивания. Одиночный результат взаимного оценивания формируется согласно оценочному листу для каждой лабораторной работы.

Итоговая оценка (**ИО**) по дисциплине «Введение в программирование» вычисляется по формуле:

**ИО = 0,7\*Н + 0,3\*ЭК**, где **Н**– оценка, накопленная за лабораторные работы первого и второго модулей, ЭК – оценка за итоговую (экзаменационную) контрольную работу.

**Н = 0,2\*Л1 + 0,2\*Т1 + 0,2\*Л2+ 0,2\*Л3 + 0,2\*Т2**

В случае, если студент получил дополнительный балл за участие в игре-олимпиаде, то он прибавляется к оценке **ИО**. Стоит отметить, что оценка **ИО** не может быть более 10 баллов.

*Округление оценок при вычислениях осуществляется до ближайшего целого.*

**План курса и организации занятий**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| № | Название раздела | Тип лекции | Работа на семинаре | Домашнее задание и элемент контроля |
|
| 1 | Алгоритмизация и программирование на языке C# | Традиционная лекция | 1) Совместное решение задач с последующим обсуждением  2) Совместное решение тестовых вопросов с последующим обсуждением | ДЗ: 1) автоматически проверяемый набор задач в GitHub Classroom;  2) чтение учебника |
| 2 | Основные алгоритмические конструкции | Традиционная лекция | 1) Командное упражнение  2) Групповая дискуссия  3) Самостоятельное решение студентами задач | Контроль – Т1  ДЗ: 1) автоматически проверяемый набор задач в GitHub Classroom;  2) чтение учебника |
| 3 | Особенности решения задач с циклами | Лекция-дискуссия | 1) Обсуждение кейсовых ситуаций в команде  2) Самостоятельное решение студентами задач  3) Совместное решение тестовых вопросов с последующим обсуждением | ДЗ: 1) автоматически проверяемый набор задач в GitHub Classroom;  2) чтение учебника |
| 4 | Структурная декомпозиция | Традиционная лекция | 1) Самостоятельное решение студентами задач и тестов  2) Командная работа | Контроль – Л1 |
| 5 | Рекуррентные соотношения. Рекурсия. | Видео-лекция | Соревновательный элемент на дополнительный балл (олимпиада по мотивам игры «Математическая карусель») | ДЗ: 1) автоматически проверяемый набор задач в GitHub Classroom;  2) чтение учебника |
| 6 | Одномерные массивы | Лекция по модели «перевернутый класс» | 1) Командное упражнение  2) Групповая дискуссия  3) Совместное решение задач с последующим обсуждением | Контроль – Л2.  ДЗ: посмотреть видео-лекцию, прочитать учебник. |
| 7 | Многомерные массивы | Видео-лекция | 1) Групповая дискуссия  2) Самостоятельное решение студентами задач  3) Совместное решение тестовых вопросов с последующим обсуждением | Контроль – Т2  ДЗ: 1) автоматически проверяемый набор задач в GitHub Classroom;  2) чтение учебника |
| 8 | Принципы разработки. Основы объектно-ориентированного программирования | Лекция-дискуссия | 1) Самостоятельное решение студентами задач  2) Совместное решение тестовых вопросов с последующим обсуждением | ДЗ: 1) автоматически проверяемый набор задач в GitHub Classroom;  2) чтение учебника |
| 9 | Отношения между классами | Традиционная лекция | Совместное решение тестовых вопросов и задач с последующим обсуждением | Контроль – Л3.  ДЗ: посмотреть видео-лекцию, прочитать учебник. |
| 10 | Абстрактные типы данных. Структуры данных | Лекция по модели «перевернутый класс» | Соревновательный элемент на дополнительный балл (олимпиада по мотивам игры «Математическая карусель») | ДЗ: 1) автоматически проверяемый набор задач в GitHub Classroom;  2) чтение учебника |

**Способы взаимодействия со студентами и организация обратной связи**

Основным каналом взаимодействия со студентами является команда в MS Teams.

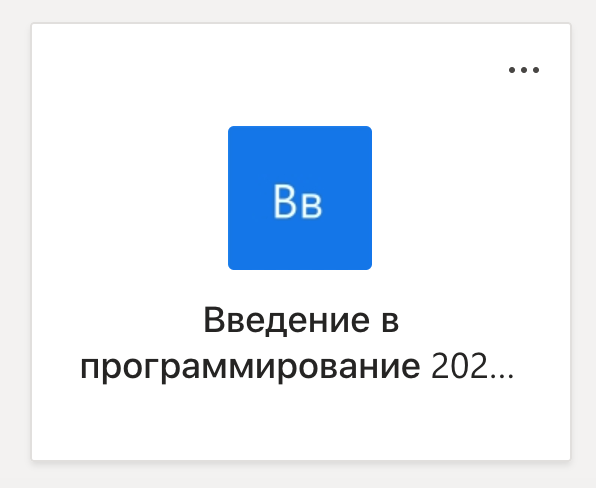


Рисунок 1 - Команда "Введение в программирование"

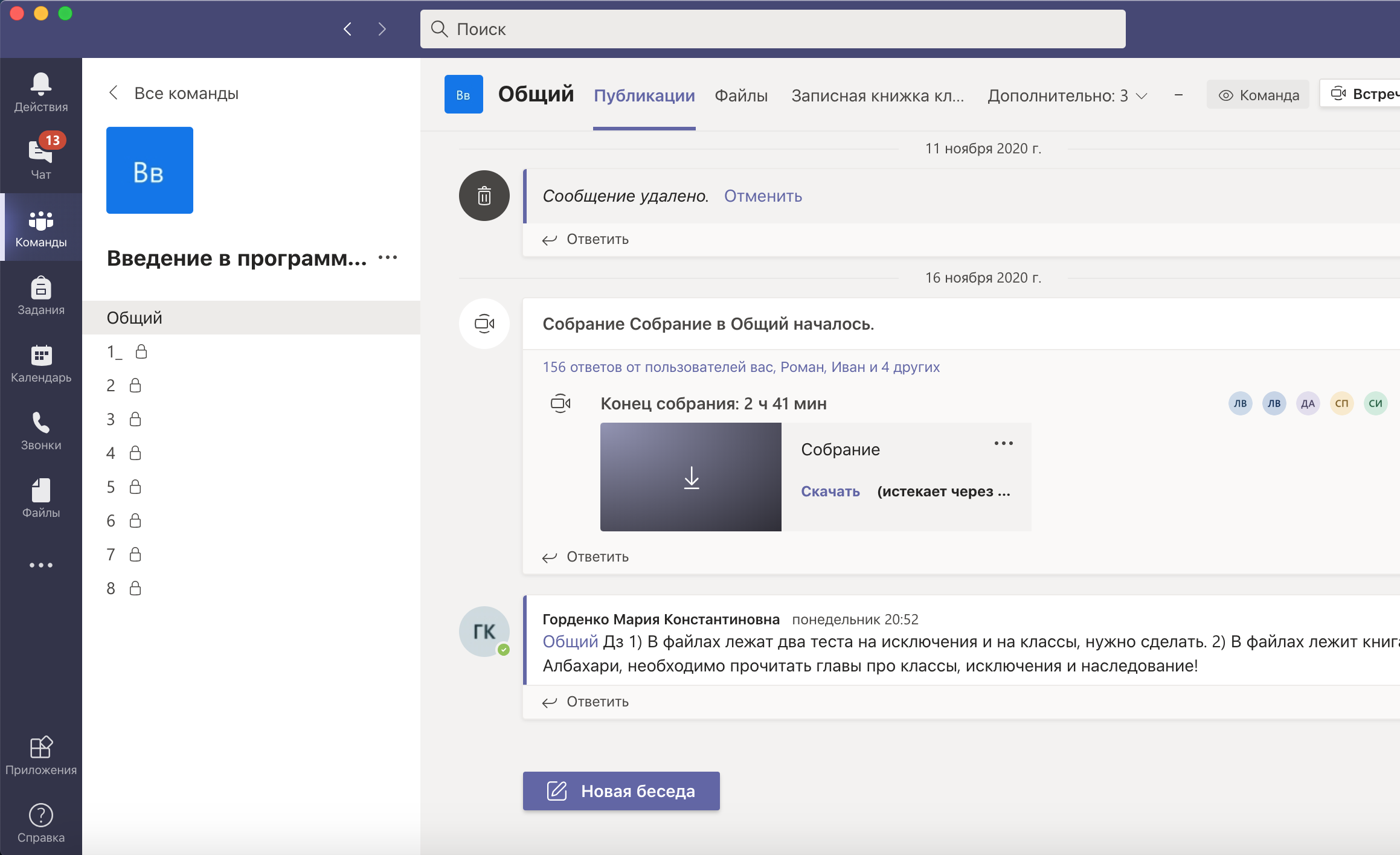


Рисунок 2 - Интерфейс MS Teams

Среди достоинств данной системы можно отметить её универсальность, т.к. она позволяет проводить занятия, хранить материалы, работать с календарем и планировщиком задач, а самое главное, что интегрирована с корпоративной почтой студентов в домене @edu.hse.ru и не требует дополнительных регистраций, а также студентам и преподавателям предоставляется бесплатно.

Кроме того, со студентами был создан Telegram-канал для оперативного решения вопросов. Однако, со студентами была договоренность решать все вопросы посредством чатов в MS Teams, но бывают ситуации, когда вопрос нужно решить очень срочно и студенты пользуются чатом Telegram.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 3 - Telegram-чат курса

Традиционным каналом взаимодействия является почта и преподаватель отвечает на вопросы по почте, однако, в основном, студенты задают все вопросы в MS Teams.

С учетом того, что в онлайн-формате обратная связь может потерять в силу того, что преподаватель не видит, а зачастую и не слышит студентов, то в ходе данного курса сбор обратной связи происходит либо путем разнообразных опросов (например, в Telegram или MS Teams) по желаемым темам, темпу обучения и т.д, либо в последние 5 минут занятия обязательно проводится рефлексия и собирается обратная связь студентов исходя из которой могут перерабатываться последующие занятия. Также студенты знают, что преподаватель открыт к диалогу и если стесняются озвучить свои проблемы, то пишут их в личные сообщения в MS Teams.

**Полученные результаты**

На текущий момент, с сентября 2020 года при таком разнообразном формате проведения занятий с большой долей вовлеченности студентов удалось несмотря на то, что курс является факультативным и посещаемость никак не учитывается при расчете оценки, сохранить посещаемость на уровне 92% (как лекционных, так и семинарских занятий). Помимо посещаемости стабильно высокий процент обучающихся, выполняющих домашние неоцениваемые задания. Скорее всего, этому причиной мгновенный фидбек. Разнообразие форм занятий и применяемых инструментов позволяет вовлечь студентов и «разговорить» их на занятии, что даже в офлайне было сделать довольно трудно.

**Описание используемых в курсе технологий**

**MS Teams** - пространство для групповой работы в Microsoft 365, где присутствуют все необходимые инструменты для проведения занятий (хранение файлов, назначение заданий, интеграция со многими сервисами, чаты, видеоконференции и т.д.). Все занятия в рамках курса «Введение в программирование» проводятся в MS Teams, также здесь хранятся все материалы курса, ведомости и т.д (см. Рисунок 1 и Рисунок 2).

**Telegram.** Как уже было описано выше, данный мессенджер используется исключительно для срочных сообщений и объявлений, т.к. не все студенты устанавливают MS Teams на свои телефоны и оперативно читают корпоративную почту. Также Telegram-чат является удобным местом для сбора обратной связи (см. Рисунок 3).

**NearPod** – сервис, который позволяет учителям разрабатывать презентации со встроенными заданиями и показывать их на любых устройствах. Преподаватель переключает слайды на собственном компьютере (можно с демонстрацией экрана), а студенты рассматривают презентацию на своих устройствах, отвечают на вопросы и выполняют задания. Ответы можно вывести на большой экран в реальном времени. Отличительной особенностью презентаций, созданных с помощью данного сервиса, является возможность предоставления студентами немедленной обратной связи в режиме реального времени. Слушатели могут отмечать уровень «понятности» как текущего слайда, так и материала в целом. Для подключения к презентации преподавателя студентам необходимо дать специальный код (написан наверху слайдов). Студентам не обязательно регистрироваться в системе. А вот преподавателю для работы придется завести собственный аккаунт. Замечательно, что если у вас уже есть готова презентация, word или pdf файл, то их можно загрузить в систему и снабдить разнообразными интерактивными вопросами. Создавать контент заново не требуется. Подробное описание здесь[[8]](#footnote-8)[[9]](#footnote-9). На Рисунок 4 приведена иллюстрация одного из возможных вариантов использования.

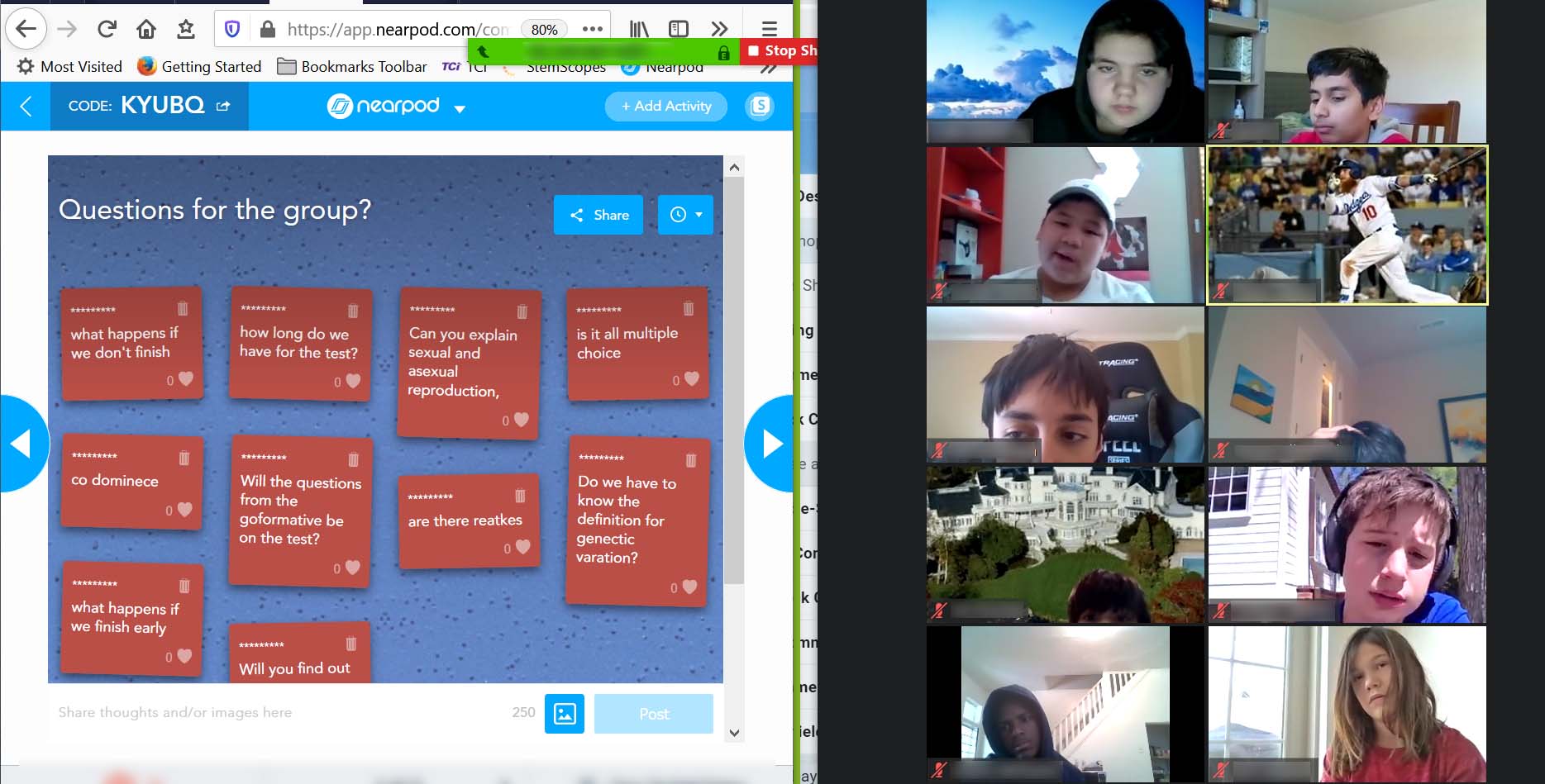


Рисунок 4 - Иллюстрация варианта использования NearPod

**PearDeck**. Данный сервис похож на сервис NearPod, однако интегрирован с Google-аккаунтами и позволяет студентам заходить под своими аккаунтами. Это довольно удобно, если для организации курса используется Google Classroom. Позволяет добавлять в обычную презентацию интерактивные элементы: тестовые вопросы, тепловые карты и т.д. Примеры вопросов можно увидеть на Рисунке 5. Подробнее можно прочитать здесь[[10]](#footnote-10).

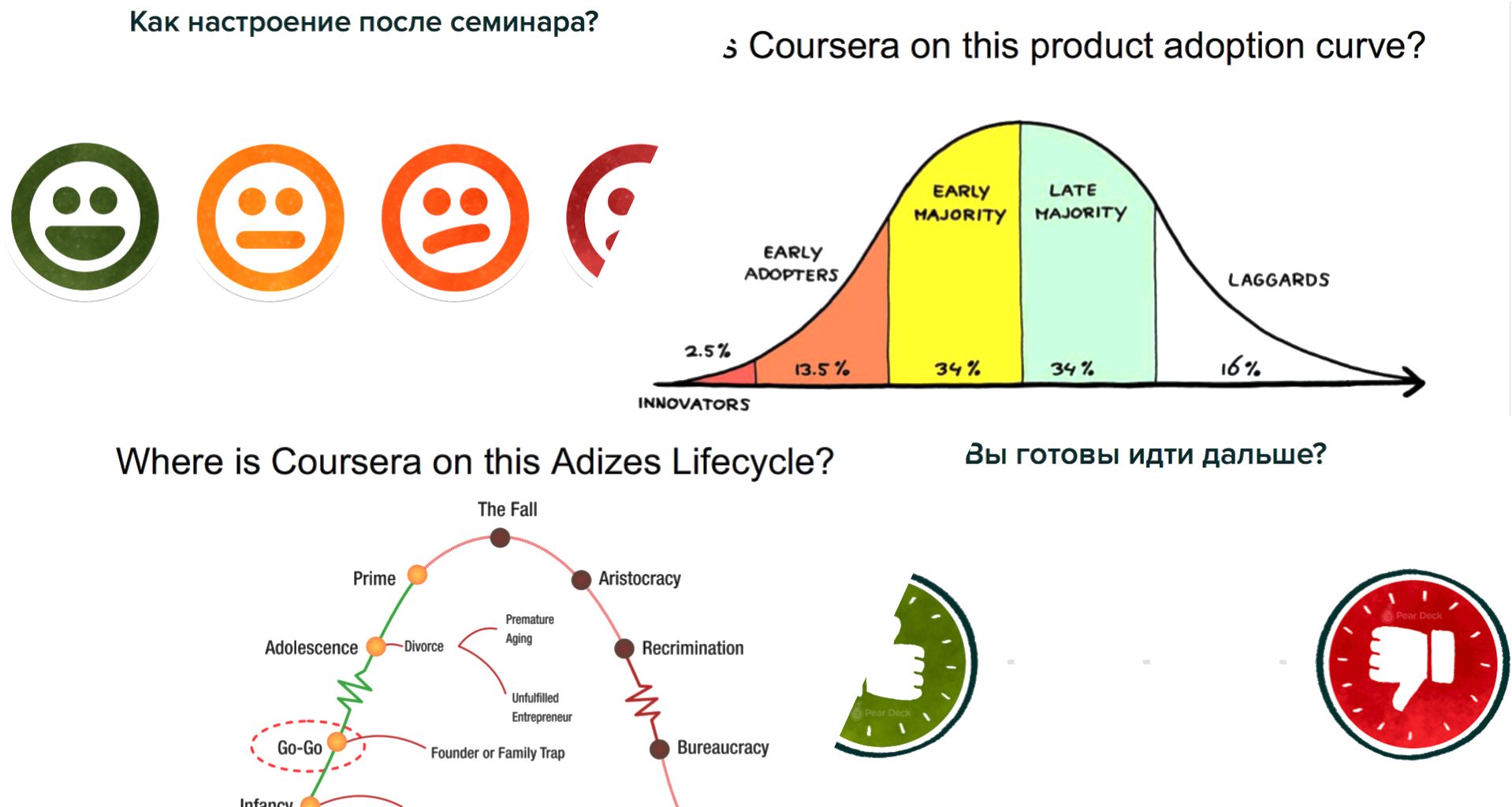


Рисунок 5 - Примеры вопросов в PearDeck

**Kahoot** – это довольно популярный сервис для создания онлайн викторин, тестов и опросов. Студенты могут отвечать на созданные тесты с планшетов, ноутбуков, смартфонов, то есть с любого устройства, имеющего доступ к Интернету. Созданные в Kahoot задания позволяют включить в них фотографии и даже видеофрагменты. Темп выполнения викторин, тестов регулируется путём введения временного предела для каждого вопроса. При желании можно ввести баллы за ответы на поставленные вопросы: за правильные ответы и за скорость. Для участия в тестировании учащиеся просто должны открыть сервис и ввести PIN-код, который указан на слайде. Пример интерфейса представлен на рисунке ниже. Более подробную информацию про данный сервис можно найти здесь[[11]](#footnote-11).

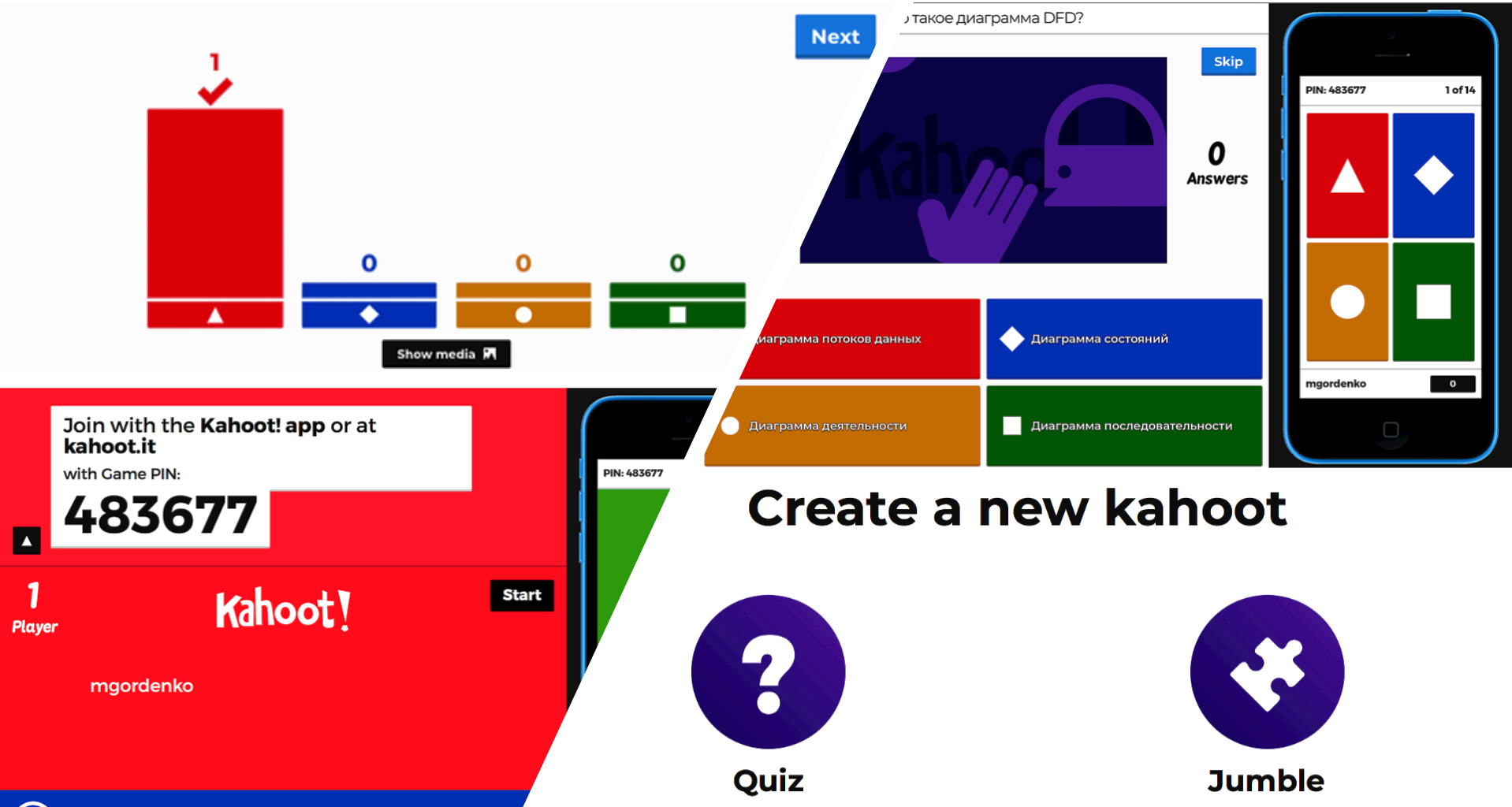


Рисунок 6 - Пример интерфейса Kahoot

**Socrative** позволяет проводить тесты в режиме реального времени, учащиеся при этом отвечают при помощи имеющихся у них устройств. Socrative — одна из самых популярных платформ для создания мобильных опросов, позволяющая создавать вопросы различных типов, и скачивать отчеты с ответами учащихся. Очевидное преимущество Socrative — учащиеся могут видеть вопросы на своих устройствах. Этим сервис отличается от Kahoot и Poll Everywhere, которые выводят вопросы только на мониторе преподавателя. В Socrative, преподаватель может добавить пояснения к каждому вопросу. Оно может появляться сразу после того, как учащийся ответил на вопрос, что делает возможным немедленную обратную связь. К тестам можно добавлять теги (метки), чтобы облегчить их поиск. Учащиеся могут выполнять тест в удобном им темпе. Есть возможность пропустить вопрос, или даже поменять ответ, прежде чем отправить тест преподавателю. В режиме Space Race учащиеся или группы учащихся могут соревноваться, кто быстрее правильно ответит на большее количество вопросов. Пример интерфейса приведен на рисунке ниже. Более подробную информацию можно прочитать по ссылке.

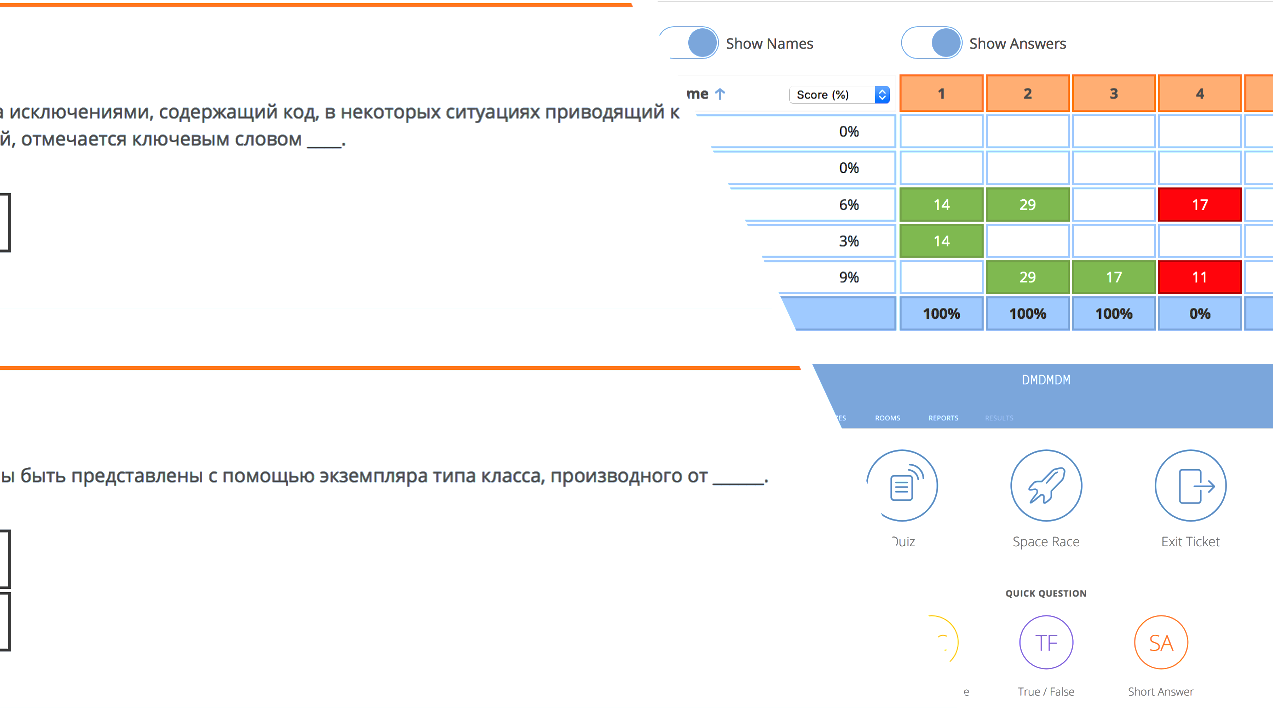


Рисунок 7 - Пример интерфейса Socrative

**PeerGrade** позволяет студентам получать обратную связь от сверстников, что открывает возможности для расширенного обучения и создает эффект «поддерживающей среды». Более того, формулирование обратной связи (peer assessment) полезно не только для того, кто ее получает, но и для того, кто оценивает. Преподавателю дается в распоряжение специальная приборная панель с рубрикатором и аналитическими функциями, чтобы видеть процесс как бы со стороны. Таким образом, с помощью данной системы происходит взаимное оценивание работ. Пример задания с рубриками оценивания приведен в приложении. Более подробно ознакомиться с функционалом можно здесь[[12]](#footnote-12).

**GitHub Classroom** предлагает веб-интерфейс, который позволяет автоматизировать создание учебных репозиториев и контроль доступа к ним. Преподаватели могут создавать домашние задания и отправлять их студентам через приватные ссылки-приглашения. Среди преимуществ стоит отметить автоматическую проверку заданий и мгновенный фидбек. На Рисунок 8 приведен формат предоставления заданий студентам. В задании содержится описание, примеры входных и выходных данных, а также шаблон кода. На Рисунок 9 приведен общий вид папки с заданиями. Кроме того, преподаватель может следить за прогрессом учеников и их ошибками (см. Рисунок 10).

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 8 - Задание в GitHub Classroom

Изображение выглядит как стол

Автоматически созданное описание

Рисунок 9 - Папка с заданиями

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 10 - Интерфейс преподавателя для отслеживания прогресса учеников

**Repl.it** – это веб-сервис, где можно писать и запускать код прямо в браузере, ничего не устанавливая к себе на компьютер. Чтобы писать код на Repl.it, не нужно скачивать и устанавливать какие-либо дополнительные инструменты. Кроме того, над проектами можно работать совместно с другими пользователями из любой страны мира, в том числе **одновременно**. Данный сервис позволяет работать с разными языками программирования, но, что важно в данном курсе, что один и тот же код студенты могут править одновременно, для этих целей данный сервис и используется. На Рисунок 11 показан интерфейс сервиса.

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 11 - Интерфейс Repl.it

**Масштабирование на другие предметы**

Описанные методики не являются новыми и уже успешно применяются в рамках разных курсов. Однако, изюминкой данной разработки является их совместное применение в рамках одного курса для поддержки образовательно процесса в онлайн, а также подобранные инструменты для их реализации.

Все приведенные методики могут быть масштабированы и на другие курсы за счет своей гибкости.

**Приложение. Пример задания в PeerGrade**

# Лабораторная работа 1

Процесс выполнения этого задания состоит из трёх частей:

1) реализация программы, согласно описанным в условии требованиям;

2) оценивание работ других студентов;

3) период «споров».

***Период реализации программы:***

После выдачи задания Вам необходимо выполнить его и загрузить архив (\*.zip) с решением задачи (полностью заархивировать решение, созданное средой разработки) до крайнего срока. В работе строго запрещается указывать ФИО, а также любую другую информацию, которая может выдать авторство работы. В случае выявления факта деанонимизации работы, работа может быть аннулирована.

***Период взаимного оценивания:***

После окончания срока, отведенного на реализацию программы, начинается период взаимного оценивания. Вам будет необходимо проверить пять работ других студентов, также выполнявших данное задание, согласно критериям оценивания. Проверка осуществляется анонимно: Вы не знаете, чью работу Вы проверяете, также, как и человек, кому принадлежит решение, не знает, кем была проверена его работа. Помимо оценки Вам необходимо указать комментарий к каждому из критериев. В случае, если Вы снижаете балл, необходимо подробно описать, за что именно была снижена оценка. Также рекомендовано писать субъективные комментарии, связанные с тонкостями программной реализации, предлагать автору работу более оптимальные на ваш взгляд решения. Снимать баллы за субъективные особенности реализации запрещено.

***Период споров:***

По окончании периода взаимного оценивания, в случае если Вы не согласны с оценкой, выставленной одним из проверяющих, Вы можете вступить с этим студентом в анонимный диалог с целью уточнения причин выставления оценки по тому или иному критерию. В случае, если проверяющий не ответил Вам, или вы не пришли к обоюдному решению об изменении оценки, Вы можете поставить флаг, и работа будет рассмотрена одним из преподавателей. Флаги, поставленные без предварительного обсуждения с проверяющим или после окончания периода выставления флагов, будут отклонены.

Также допустима перепроверка Вашей работы преподавателем. В таком случае оценки других проверявших работу не учитываются.

**Дедлайн загрузки работы:** 7 декабря 23:59

**Дедлайн проверки:** 10 декабря 23:59

**Дедлайн обсуждения оценок:** 12 декабря 23:59

**Дедлайн выставления флагов:** 13 декабря 23:59

**Возможность поздней сдачи работы в этом задании предоставляться не будет.**

**Оценивание**:

Оитог = 0,8 \* Озадание + 0,2 \* Опроверки , где Озадание – неокруглённая десятибалльная оценка за решение задания выставленная проверяющими с учётом возможной перепроверки преподавателем, а Опроверки неокруглённая десятибалльная оценка, выставленная студентами, чьи работы вы проверяли. Также в случае, если преподаватель обнаружит, что Вы проверили работы некачественно к Вам могут быть применены санкции в виде штрафа до 3 десятичных баллов от Оитог (в таком случае оценка вычисляется по формуле Оитог = Озадание – Штраф).

**Алгоритм работы приложения:**

Пользователь вводит числа 𝑛, 𝑚, 𝑚𝑎𝑥, 𝑚𝑖𝑛:

* 𝑛 – длина массива массивов
* 𝑚 – максимальная длина подмассива
* 𝑚𝑖𝑛 – минимальный элемент подмассива
* 𝑚𝑎𝑥 – максимальный элемент подмассива

Необходимо сформировать массив массивов 𝑎𝑟𝑟, который содержит 𝑛 одномерных вещественных массивов со случайной длиной от 1 до 𝑚. Каждый одномерный массив (подмассив) должен быть заполнен случайными вещественными числами от 𝑚𝑖𝑛 до 𝑚𝑎𝑥. По массиву 𝑎𝑟𝑟 сформировать два массива массивов, содержащих только целые и только дробные части элементов исходного массива 𝑎𝑟𝑟. Все массивы вывести на экран.

Для этого реализуйте и используйте следующие методы:

* **double[][] CreateArray(int n, int m, int min, int max)** - создает массив массивов длины

𝑛. Каждый элемент массива массивов имеет случайную длину от 1 до 𝑚

(включительно), и заполняется случайными вещественными числами от 𝑚𝑖𝑛 до 𝑚𝑎𝑥;

* **void SeparateArray(double[][] arr, out int[][] integervalues, out double[][]**

**fractionalvalues)** – метод, который по массиву массивов с вещественными элементами формирует два массива массивов. В первый массив массивов 𝑖𝑛𝑡𝑒𝑔𝑒𝑟𝑣𝑎𝑙𝑢𝑒𝑠 помещаются все целые части элементов исходного массива 𝑎𝑟𝑟, а во второй 𝑓𝑟𝑎𝑐𝑡𝑖𝑜𝑛𝑎𝑙𝑣𝑎𝑙𝑢𝑒𝑠 – дробные. Порядок следования элементов, их целых и дробных частей во всех массивах сохраняется.

* **две перегрузки void PrintJaggedArray(double[][] arr)** и **void PrintJaggedArray(int[][] arr)** – метод для печати массива массивов. Элементы исходного массива печатать на новой строке, а элементы подмассивов через пробел (см. пример ниже).

Снабжайте все выводы на экран вспомогательными текстовыми сообщениями.

Нужно реализовать повтор решения. Также можно реализовывать свои дополнительные методы для решения данной задачи.

**Дополнительные требования (критерии оценивания):**

1. Для проверки в систему PeerGrade должен быть загружен архив с решением. Ожидается, что проверка работы будет проводиться, в среде разработки Visual Studio 2019, поэтому в случае выполнения задания с использованием другой среды разработки настоятельно рекомендуется проверить возможность открытия и запуска проекта в этой среде разработки.
2. Текст программы должен быть отформатирован согласно кодстайлу языка C#[[13]](#footnote-13). Для автоматического форматирования в среде Visual Studio достаточно нажать Ctrl+K, D. Идентификаторы должны соответствовать соглашению о именовании C#[[14]](#footnote-14). Основным требованием, которое требуется соблюдать в данном задании – это написание имён локальный переменных с использованием camelCasing, а при именовании методов и типов PascalCasing.
3. Программа не должна завершаться аварийно (или уходить в бесконечный цикл) при любых входных данных. При некорректных входных данных программа должна выводить сообщение об ошибке и запрашивать ввод заново.
4. Программа должна быть декомпозирована. Каждый из логических блоков должен быть выделен в отдельный метод. Не строго, но желательно, чтобы каждый метод по длине не превышал 40 строк.
5. Интерфейс программы должен быть понятен. Пользователю должны выводиться подсказки о возможных дальнейших действиях и иные необходимые сообщения. Предполагается, что для успешного использования программы не требуется обращения к исходному коду программы.
6. Текст программы должен быть документирован. Необходимо писать, как комментарии перед методами, так и комментарии, поясняющие написанный внутри метода код. Названия переменных и методов должны быть на английском языке и отражать суть хранимых значений / выполняемых действий.
7. Для получения отличной оценки более 8 студенту необходимо реализовать дополнительный функционал.
8. Также оценивается общее впечатление, которое производит работа (как с точки зрения пользовательского интерфейса, так и с точки зрения написания кода программы). Эта часть оценки остаётся на усмотрение проверяющего.

Таким образом выглядят рубрики в системе при оценке работ других студентов (Рисунок 12):

Изображение выглядит как текст

Автоматически созданное описание

Рисунок 12 - Рубрики PeerGrade

1. <https://teams.microsoft.com> [↑](#footnote-ref-1)
2. <https://nearpod.com> [↑](#footnote-ref-2)
3. <https://www.peardeck.com> [↑](#footnote-ref-3)
4. <https://kahoot.it> [↑](#footnote-ref-4)
5. <https://www.socrative.com> [↑](#footnote-ref-5)
6. <https://repl.it/> [↑](#footnote-ref-6)
7. <https://classroom.github.com> [↑](#footnote-ref-7)
8. <http://didaktor.ru/nearpod-prilozhenie-dlya-effektivnogo-mobilnogo-obucheniya/> [↑](#footnote-ref-8)
9. <https://newtonew.com/app/nearpod-teach-with-mobile> [↑](#footnote-ref-9)
10. <https://newtonew.com/app/novyj-instrument-dlja-uchitelej-prostoe-i-vovlekatelnoe-prilozhenie-dlja-interaktivnyh-prezentacij> [↑](#footnote-ref-10)
11. <https://rosuchebnik.ru/material/kahoot-servis-dlya-organizatsii-onlayn-viktorin-testov-i-oprosov/> [↑](#footnote-ref-11)
12. <https://foi.hse.ru/data/2017/12/28/1160691117/Приложение%205%20-%20интерфейс%20peergrade.pdf> [↑](#footnote-ref-12)
13. <https://docs.microsoft.com/en-us/dotnet/csharp/programming-guide/inside-a-program/coding-conventions> [↑](#footnote-ref-13)
14. https://docs.microsoft.com/en-us/dotnet/standard/design-guidelines/naming-guidelines [↑](#footnote-ref-14)