Муниципальное бюджетное общеобразовательное учреждение г. Иркутска средняя общеобразовательная школа №35

**Игры на языке программирования Python**

Работу выполнил: **Маньков Артём**

учащийся 8 «А» класса

Руководитель: **Белоусова Г.В.**

учитель информатики

Иркутск, 2024 год

**Содержание**

1. [Введение 3](#_Toc166896408)
2. [Теоретическая часть 5](#_Toc166896409)
3. [Языки программирования 5](#_Toc166896410)
4. [История программирования 5](#_Toc166896411)
5. [Что нужно знать о Python 8](#_Toc166896412)
6. [Pascal – это? 9](#_Toc166896413)
7. [Сравнение Python и Pascal как часть образования 10](#_Toc166896414)
8. [Практическая часть 12](#_Toc166896415)
9. [Заключение 13](#_Toc166896416)
10. [Список литературы 14](#_Toc166896417)
11. [Приложения 15](#_Toc166896418)

Введение

В современном мире информационных технологий компьютерные игры играют значительную роль в повседневной жизни многих людей. Они не только предоставляют возможность развлечения и отдыха, но и являются инструментом для обучения, развития логического мышления и креативности. В этом контексте изучение создания игр на языках программирования становится не только увлекательным и интересным занятием, но и важной частью образования, особенно для молодежи.

Python, наиболее популярный и доступный язык программирования, прекрасно подходит для разработки игр благодаря своей простоте, эффективности и обширным возможностям библиотек.

Однако, не следует забывать и о других языках программирования, таких как Pascal. Pascal долгое время является основным языком программирования для изучения в рамках школьной программы, благодаря своей структурности и строгому синтаксису.

Во время выполнения проекта мы изучим процесс создания компьютерных игр с использованием языка программирования Python. Мы рассмотрим основные концепции разработки игр, а также создадим простой игровой проект для демонстрации применения полученных знаний. Предполагается, что изучение данной темы позволит не только приобрести практические навыки программирования, но и стимулирует творческое мышление и логическое мышление. Результаты данного исследования могут быть полезны для учителей информатики, школьников и всех, кто интересуется миром программирования и разработки игр.

Актуальность - растущая популярность Python, особенно среди начинающих разработчиков и школьников, делает изучение этого языка интересным и востребованным. Игры являются способом не только изучать программирование, но и развивать логическое мышление, креативность и умение работать в команде, что делает эту тему актуальной для образовательных целей.

Цель проекта - изучение процесса создания компьютерных игр с использованием языка программирования Python и оценка их влияния на развитие когнитивных навыков учащихся.

Задачи:

1. Ознакомиться с историей развития языков программирования
2. Подробнее изучить информацию о языках программирования Python и Pascal
3. Провести сравнительную характеристику Python и Pascal
4. Выбрать язык программирования для разработки
5. Выбрать тему для своей игры
6. Написать код для игры

Гипотеза - изучение и создание игр на Python способствует более глубокому пониманию основ программирования у начинающих разработчиков и может стать мотивацией для изучения информатики в школе.

# Теоретическая часть

## Языки программирования

Язык программирования — формальный язык, предназначенный для записи компьютерных программ. Язык программирования определяет набор лексических, синтаксических и семантических правил, определяющих внешний вид программы и действия, которые выполнит исполнитель под её управлением.

## История программирования

Мы привыкли к тому, что программирование неразрывно связано с созданием первых ЭВМ (электронно-вычислительных машин), которые появились в середине 1940-х годов. Однако это мнение не совсем точное.

Прообразом привычного компьютера стала разностная машина английского математика Чарльза Бэббиджа. Работу над ней он начал ещё в 1822 году и надеялся, что однажды она заменит людей, которые регулярно допускали ошибки в навигационных, астрономических и математических таблицах.

Бэббидж ничего не знал о микросхемах и транзисторах, но ему удалось придумать аппарат, который мог функционировать и производить вычисления. Всех поставленных целей Бэббиджу достичь не удалось, но он сделал главное – указал направление развития компьютерного мира.

Возможно, основной ошибкой этого учёного, оставившего след в истории, было использование десятичной системы счисления и слишком узкое представление о потенциале своего аппарата. Гораздо больше преимуществ этой машины увидела Ада Лавлейс (настоящее имя Августа Ада Байрон) – дочь английского поэта Джорджа Байрона, которая с детства интересовалась математикой.

Когда она переводила лекцию Чарльза Бэббиджа о его разностной машине, то внесла некоторые существенные изменения. Прежде всего, Ада стала называть её аналитической. Она считала, что в будущем такой аппарат сможет не только производить математические преобразования, но и обрабатывать ту информацию, которую в него заложат.

Но, что самое интересное, именно Ада написала первую в истории компьютерную программу. Это был алгоритм вычисления чисел Бернулли (последовательность рациональных чисел, которые возведены в одну и ту же степень). Ада ввела такие понятия, как “цикл” и “ячейка”, без которых уже сложно представить компьютерную лексику, и описала необходимые элементы любой вычислительной машины:

* устройство для кодирования цифровых данных (перфокарты);
* память для хранения числовых значений;
* блок управления, через который пользователь сообщает машине, какие задачи выполнять;
* “мельница”, ответственная за выполнение расчётов;
* устройство для просмотра результатов.

Несмотря на такое подробное описание, первые электрические ЭВМ появились лишь спустя 100 лет, так же, как и прототип первого языка программирования высокого уровня.

Итак, в конце Второй мировой войны был изобретен первый электронный вычислитель (ЭНИАК, сокращение от англ. Electronic Numerical Integrator and Computer). Однако он являлся лишь электронным цифровым вычислителем общего назначения.

Программы для такого вычислителя нужно было писать на понятном ему языке, то есть использовать машинный код. Кроме того, для каждого ЭНИАК требовалась новая программа, написание которой отнимало много времени и сил.

Но это направление активно развивалось, и уже в 1947 Кэтлин Бут, британско-канадский учёный, разработала язык ассемблера – автокода для первых компьютеров. Эту идею подхватил другой английский учёный, который в 1948 году написал код для более современных ЭВМ.

Таким образом, можно сказать, что базовые машинные коды были первым поколением языков программирования, а язык ассемблера – вторым.

Настоящим прорывом стала разработка первого компилятора – программы, которая автоматически переводила текст, написанный на языке программирования, в машинный код. Его создала Грейс Мюррей Хоппер в 1952 году.

Самым первым языком программирования высокого уровня считается Планкалкюль (нем. Plankalkül — «запланированные вычисления»), который был создан в 1948 году немецким учёным Конрадом Цузе специально для вычислительных устройств его собственного изобретения Z4.

К сожалению, все предыдущие модели этого устройства были разрушены в 1944 во время бомбардировок Берлина, а сам разработчик находился в некоторой изоляции от научного сообщества того времени, поэтому его труды увидели свет лишь в начале 1970-х годов. А сам язык никогда не применялся для других ЭВМ.

Fortran – язык, разработанный в 1957 году, который стал альтернативой ассемблеру и даже в наше время применяется для инженерных расчётов, написания программ и библиотек подпрограмм.

Период с начала 1960-х и до конца 1980-х ознаменовался активным развитием компьютерных технологий, а вместе с этим и языков программирования. Так, ещё в 1964 году Томас Курц и Джон Кемени, профессора Дартмутского колледжа, разработали язык BASIC.

Примерно в это же время швейцарский специалист в области информатики Никлаус Вирт работал над созданием другого легендарного языка – Pascal. А в 1969 году появилась операционная система UNIX, которую разработал Кен Томпсон.

Томпсон совместно со своим коллегой Деннисом Ритчи работал над операционной системой Multics, но проект не был завершён. В итоге основным направлением стала разработка UNIX – системы, которая в корне изменила представление пользователей о компьютерных технологиях и оказала влияние на развитие всей сферы IT.

Другим значимым достижением Томпсона и Ритчи стало создание языка программирования В и на его базе – языка С. Именно Деннис Ритчи считается разработчиком языка С, который спустя почти 50 лет продолжает использоваться для разработки операционных систем и приложений. Синтаксис этого языка стал основой для более поздних языков Java, C++, Objective-C и C#

В 1980-е годы мир компьютерных кодов формировался и постоянно расширялся.

В 1989 году Тим Бернерс-Ли, английский физик, представил всемирную паутину, а в следующие 2 десятилетия основной упор был уже не на разработку операционных систем, а на веб-программирование.

Инновационные для 1990-х и начала 2000-х годов языки программирования стали более простыми и доступными. Они часто интегрированы в веб-страницы и закодированы таким образом, чтобы облегчить цепочку простых команд.

* Python создан в 1991 году
* Ruby изобретён в 1993 году
* PHP написан в 1994 году
* JavaScript создан в 1995 году
* Java создан в 1996 году
* С# разработан в 2001 году

Сейчас любое обучение программирование невозможно представить без изучения этих языков, но тогда это был настоящий переворот. Без PHP на сайтах не было бы форм для заполнения и волшебной кнопки “отправить”. На Java написаны известные мировые сайты Amazon, eBay и LinkedIn, а также Одноклассники и сайт РЖД. А Python используют такие крупные компании, как Google, Instagram, Spotify, Dropbox и многие другие.

Программирование больше не считается скучным и исключительно мужским занятием. Основная тенденция последних 10 лет заключается в совершенствовании существующих знаний в области программирования, а не создание новых языков. Мы видим, что языки, разработанные в 90-ые годы постоянно адаптируются под новые IT-задачи.

## Что нужно знать о Python

В 1982 году Гвидо ван Россум окончил университет и попал в команду разработчиков института CWI, где до 1986 года занимался проектированием языка ABC — прототипа Python. ABC задумывался как инструмент для пользователей, которые до этого не программировали и не разбирались в устройстве компьютеров. Должен был получиться удобный язык с простым синтаксисом, на котором легко учиться писать программы.

В 1987 году проект ABC закрылся. По мнению Гвидо, главная причина заключалась в отсутствии доступного интернета — язык медленно распространялся и не получал оперативной обратной связи от пользователей. Из-за этого команда не добавляла улучшения, которые учитывали бы потребности разработчиков. Сами программисты не могли присоединиться к сообществу и поучаствовать в развитии проекта. Язык ABC опередил своё время и мог стать заменой Python.

В 1986 году Гвидо оставался сотрудником CWI, но перешёл из ABC в проект Amoeba — там он разрабатывал операционную систему для крупных организаций. Amoeba объединяла сеть пользовательских компьютеров в единую станцию и оптимизировала их совместную работу над производительными задачами.

В 1989 году системе Amoeba не хватало языка сценариев, поэтому Гвидо ван Россум планировал мини-проект: собирался написать язык программирования на основе наработок ABC. Гвидо сделал первые наброски и вскоре показал коллегам прототип будущего Python.

Первый прототип состоял из простой виртуальной машины, парсера и среды выполнения. В нём также присутствовал базовый синтаксис, операторы, словари, списки, строки и небольшое количество других типов данных. Будущий Python предлагал гибкую модель расширяемости — то есть, помимо стандартных возможностей, каждый программист мог самостоятельно добавить в систему нужные типы объектов.

Прототип Python понравился разработчикам CWI, и многие сразу включились в процесс: начали использовать язык для внутренних проектов и помогли доработать код.

20 февраля 1991 года Гвидо ван Россум создал дистрибутив и опубликовал код языка Python.

29 июня 1994 года на форуме вышла статья «Если бы Гвидо сбил автобус?». Публикация затронула проблему зависимости Python-сообщества от решений Гвидо ван Россума — автор поделился мнением о том, что крупные компании опасаются использовать технологии, которые привязаны к одному человеку.

В 1995 году Python Software Foundation — некоммерческой организации, которая должна была отвечать за защиту и развитие языка Python. У этой организации появилось несколько руководителей.

Вторая версия Python появилась в 2000 году, а 3 декабря 2008 года появилась третья версия Python, которая устраняла критические неисправности и перерабатывала архитектуру языка. Из-за большого количества изменений вторая и третья версии оказались не полностью совместимыми и до 2020 года развивались по отдельности. С конца 2020 года официальное Python-сообщество поддерживает только третью версию. Восемь лет компании создавали программы, библиотеки и приложения, которые после обновления пришлось полностью или частично переводить на Python 3.0.

В 2021 году языку Python исполнилось 30 лет. За это время он превратился из домашнего прототипа в мировой язык программирования. Самое важное — с 2019 года Python полностью независим от основателя и развивается силами сообщества.

## Pascal – это?

Pascal — это универсальный язык программирования, отличающийся строгой структурой и типизацией переменных. Был разработан швейцарским ученым Никлаусом Виртом в 1970 году на базе языка Алгол-68 для обучения студентов структурному программированию. Некоторые диалекты Pascal ограниченно применяются в промышленности и разработке приложений.

Язык Pascal относится к семейству императивных, то есть написанная на нем программа представляет собой набор последовательных команд. При этом: команды выполняются строго последовательно; результат выполнения команды записывается в память; каждая последующая команда использует данные, полученные при исполнении предыдущей.

Также Pascal является языком со строгой типизацией. Различным конструкциям, из которых состоит программа (модулям, функциям, переменным и т.д.), присвоены четко определенные свойства. Это позволяет согласовать работу отдельных частей программного кода и уменьшить вероятность ошибок, вызываемых совместным применением логически несовместимых значений.

Разработчик Pascal — швейцарский ученый Никлаус Вирт. Это один из наиболее заметных специалистов по информатике своего времени, создатель многих языков программирования. В конце 60-х годов прошлого века он участвовал в комиссии по разработке стандарта языка Алгол-68. Часть этих наработок он использовал затем при создании Pascal. Название новому языку Вирт дал в честь французского ученого, механика и изобретателя Блеза Паскаля, создавшего одно из первых механических счетных устройств.

Главной целью Вирта было создание удобного инструмента для обучения студентов структурному программированию.

Дальнейшее развитие Pascal привело к появлению многих диалектов этого языка, созданных как самим Виртом, так и сторонними разработчиками:

В 1983 году специалистами фирмы Borland был разработан диалект Turbo Pascal и интегрированная среда разработки с отладчиком и встроенными библиотеками на его основе. Он оказался очень популярным за счет быстрой компиляции и исполнения кода, поэтому стал основой для программирования прикладных приложений на популярных микрокомпьютерах того времени.

В 1986 году на базе Pascal появился объектно-ориентированный язык программирования Delphi с собственной средой разработки. Он в целом сохранил простоту и интуитивность своего прародителя и стал распространенным инструментом прикладного программирования. Однако, в отличие от ориентированного на обучение Pascal, язык и среда Delphi стали использоваться и для решения практических задач — например, программирования промышленных контроллеров.

Важным этапом стало появление свободных версий языка Pascal. Например, Free Pascal и GNU Pascal представляют собой диалекты Object Pascal со свободно распространяемым компилятором. Благодаря ему написанные на этом языке приложения можно было запускать на различных платформах под управлением разных операционных систем.

Еще одна важная характеристика языка Pascal — его процедурность и структурированность. Это означает, что вся программа состоит из блоков, которые находятся по отношению к ней и друг к другу в строгой подчиненности. При этом отдельные последовательно выполняемые блоки можно объединять в подпрограммы.

Pascal стал одним из первых языков программирования такого типа. Это обусловлено тем, что в 60–70-е годы прошлого века сложность компьютерных вычислений достигла такого уровня, что писать программы неструктурированным языком стало неудобно и долго. Более того, программы без структуры (так называемый спагетти-код) было сложно читать, анализировать и тестировать, что вело к большому количеству ошибок. По мнению самого Вирта, разработанный им язык позволяет обучить студентов не просто программировать, а писать простой и удобный для понимания код.

## Сравнение Python и Pascal как часть образования

Python и PascalABC - два популярных языка программирования, каждый из которых обладает своими особенностями и предназначением. Сравнительный анализ этих языков позволяет выявить их сходства и различия, что помогает определиться с выбором подходящего инструмента для конкретных задач и целей. Рассмотрим основные характеристики Python и PascalABC, чтобы понять, в чем заключаются их основные отличия.

Синтаксис:

* Python: известен своим простым и лаконичным синтаксисом. Он использует пробелы для обозначения блоков кода (в отличие от фигурных скобок или ключевых слов, как в PascalABC).
* PascalABC: имеет более строгий синтаксис с использованием ключевых слов и разделителей, таких как точка с запятой.

Типы данных:

* Python: является динамически типизированным языком, что означает, что типы данных определяются автоматически при выполнении программы.
* PascalABC: относится к статически типизированным языкам, где типы данных должны быть объявлены до их использования.

Применение:

* Python: широко используется во многих областях, таких как веб-разработка, анализ данных, искусственный интеллект, научные вычисления и др.
* PascalABC: часто используется в учебных заведениях, особенно в России, для обучения основам программирования.

Среда разработки:

* Python: множество сред разработки доступны для Python, таких как PyCharm, Visual Studio Code, Jupyter Notebook и др.
* PascalABC: обычно используется в среде PascalABC.NET, которая предоставляет удобное окружение для разработки на языке PascalABC.

Сообщество и экосистема:

* Python: обладает огромным сообществом разработчиков, богатой экосистемой библиотек и фреймворков, что делает его привлекательным для быстрой разработки.
* PascalABC: в сравнении с Python экосистема PascalABC менее развита, но существует некоторое количество библиотек и инструментов для разработки.

Таким образом, я могу сделать вывод, что, если целью является предоставление широких возможностей в будущем и изучение современных технологий, то Python может быть более подходящим выбором. Исходя из вышеперечисленных доводов, была сформирована таблица плюсов и минусов каждого из языков. (см. прилож.)

# Практическая часть

В данной главе представлен результат практического исследования, направленного на изучение языка программирования Python через разработку игры Крестики-нолики. В процессе создания игры были применены основные концепции и принципы программирования на Python, что позволило глубже понять его синтаксис и особенности.

Перед тем как приступить к разработке игры, был проведен анализ языков программирования, применяемых в образовательных учреждениях. Исторически школы использовали Pascal, однако Python, с его простым и понятным синтаксисом, показался наиболее предпочтительным для изучения в школьной среде. Его читаемость и гибкость делают его отличным выбором для начинающих программистов, позволяя сосредоточиться на освоении основных концепций программирования, а не на синтаксисе.

Целью практической части проекта было создание игры с использованием языка Python. Для этого были выполнены следующие шаги:

1. Выбор темы и концепции игры: я решил разработать простую игру на основе консольного ввода-вывода, чтобы сосредоточиться на основах языка.
2. Написание кода: я написал код игры с использованием языка Python. В процессе написания игры были применены основные функции языка, такие как условия, циклы, функции и массивы.
3. Тестирование и отладка: после завершения написания кода игры было проведено тестирование для проверки ее функциональности и выявления возможных ошибок. Обнаруженные ошибки были устранены в процессе отладки.

Написанная игра представляет собой окно консоли с выведенным массивом чисел 3x3. Для того, чтобы сделать ход нужно ввести номер свободной клетки, после ввода номера клетки выводится новый массив, с замененным номером клетки на X или O (в зависимости от очередности хода игроков). Цель игры состоит в том, чтобы выстроить на игровом поле три своих символа (крестика или нолика) в ряд по вертикали, горизонтали или диагонали. В случае заполнения всего поля символами без выигрышной комбинации одного из игроков, игра сообщит о ничье (Код игры см. в прилож.)

# Заключение

В результате проекта по изучению основ программирования на языке Python при помощи разработки игры "Крестики-нолики" можно сделать следующие выводы: Python является наиболее предпочтительным языком программирования для изучения в школе из-за его простого и понятного синтаксиса, широкого применения и большой популярности. Создание игры позволило применить теоретические знания в практике, что углубило понимание основных концепций языка. Проект подтвердил эффективность проектного подхода к обучению, где теория сочетается с практикой, и показал практическую значимость полученных знаний. Таким образом, выполнение проекта не только улучшило навыки программирования на Python, но и подтвердило важность использования практических задач в обучении.

# Список литературы

1. Босова, Л. Л. Информатика. [Текст] / Л. Л. Босова — 7-е изд.. — Москва: Бином, 2018 — 155 c.
2. Бабаскин А.П. Краткая история Python / Бабаскин А.П. [Электронный ресурс] // Skillbox : [сайт]. — URL: https://skillbox.ru/media/code/kratkaya-istoriya-python/ (дата обращения: 02.03.2024).
3. Кольцов, Д. М. Справочник PYTHON. Кратко, быстро, под рукой [Текст] / Д. М. Кольцов — 1-е изд.. — Санкт-Петербург: Наука и техника, 2021 — 288 c.
4. Pascal / [Электронный ресурс] // Skillfactory : [сайт]. — URL: https://blog.skillfactory.ru/glossary/pascal/ (дата обращения: 02.03.2024).

# Приложения

Сравнительная таблица плюсов и минусов языков программирования Python и Pascal:

Таблица № 1

|  |  |  |
| --- | --- | --- |
| **Характеристика** | **Python** | **Pascal** |
| Синтаксис | (+) Простой и лаконичный синтаксис | (+) Строгий синтаксис, полезный для обучения |
|  | (+) Использование пробелов для обозначения блоков | (-) Более сложный синтаксис с ключевыми словами и разделителями |
| Типы данных | (+) Динамическая типизация | (+) Статическая типизация обеспечивает безопасность типов |
|  | (-) Возможны ошибки из-за динамической типизации | (-) Необходимость явного объявления типов |
| Применение | (+) Широкое применение: веб-разработка, анализ данных, ИИ и др. | (-) Ограниченное применение за пределами образовательных целей |
| Среда разработки | (+) Множество сред разработки: PyCharm, VS Code и др. | (+) PascalABC.NET предоставляет удобное окружение для разработки |
| Сообщество и экосистема | (+) Огромное сообщество разработчиков, богатая экосистема библиотек и фреймворков | (-) Менее развитое сообщество и экосистема, ограниченное количество библиотек |
| Уровень сложности | (+) Подходит для начинающих благодаря простоте синтаксиса | (+) Полезен для обучения благодаря строгому синтаксису и структуре |
| Возможности в будущем | (+) Широкие возможности для развития и изучения современных технологий | (-) Меньше возможностей для использования в реальных проектах вне учебных заведений |

## Код написанной игры

print("Представлено игровое поле для игры в крестики-нолики:")

def allgame():

 pole = [[1, 2, 3],

 [4, 5, 6],

 [7, 8, 9]]

 def console():

 # Вывод массива

 for row in pole:

 for element in row:

 print(element, end=' ')

 print()

 def proverkaX():

 # Проверка строк

 for row in pole:

 if row.count("X") == 3:

 print("Победа! Крестики выиграли!")

 return True

 # Проверка столбцов

 for col in range(3):

 if pole[0][col] == pole[1][col] == pole[2][col] == "X":

 print("Победа! Крестики выиграли!")

 return True

 # Проверка диагоналей

 if pole[0][0] == pole[1][1] == pole[2][2] == "X" or pole[0][2] == pole[1][1] == pole[2][0] == "X":

 print("Победа! Крестики выиграли!")

 return True

 return False

 def proverkaO():

 # Проверка строк

 for row in pole:

 if row.count("O") == 3:

 print("Победа! Нолики выиграли!")

 return True

 # Проверка столбцов

 for col in range(3):

 if pole[0][col] == pole[1][col] == pole[2][col] == "O":

 print("Победа! Нолики выиграли!")

 return True

 # Проверка диагоналей

 if pole[0][0] == pole[1][1] == pole[2][2] == "O" or pole[0][2] == pole[1][1] == pole[2][0] == "O":

 print("Победа! Нолики выиграли!")

 return True

 return False

 def proverkaDraw():

 # Проверка занятости всех полей

 for row in pole:

 if any(cell != "X" and cell != "O" for cell in row):

 return False

 print("Ничья!")

 return True

 console()

 while not proverkaX() and not proverkaO():

 while True:

 xodX = int(input("Введите номер клетки (ход крестиков): "))

 if xodX >= 1 and xodX <= 9:

 tempX = xodX - 1

 row\_index = tempX // 3

 col\_index = tempX % 3

 if pole[row\_index][col\_index] != "X" and pole[row\_index][col\_index] != "O":

 pole[row\_index][col\_index] = "X"

 break

 else:

 print("Неверный номер клетки.")

 else:

 print("Неверный номер клетки")

 console()

 console()

 if proverkaX() or proverkaO() or proverkaDraw():

 break

 while True:

 xodO = int(input("Введите номер клетки (ход ноликов): "))

 if xodO >= 1 and xodO <= 9:

 tempO = xodO - 1

 row\_index = tempO // 3

 col\_index = tempO % 3

 if pole[row\_index][col\_index] != "X" and pole[row\_index][col\_index] != "O":

 pole[row\_index][col\_index] = "O"

 break

 else:

 print("Неверный номер клетки.")

 else:

 print("Неверный номер клетки")

 console()

 console()

 if proverkaX() or proverkaO() or proverkaDraw():

 break

 input("Нажмите Enter для продолжения...")

while True:

 allgame()

## Карта самооценки

Фамилия Имя участника проекта: Маньков Артём

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Активно: выполнял свою работу самостоятельно, вовремя (2 балла)** | **Не очень активно: с помощью родителей, учителя, товарищей, не всегда вовремя (1 балл)** | **Пассивно: часто не справлялся вовремя, не выполнял свою часть работы (0 баллов)** |
| Постановка проблемы проекта | 2 |  |  |
| Формулирование темы проекта | 2 |  |  |
| Формулирование целей, задач, ожидаемых результатов | 2 |  |  |
| Составление плана работы | 2 |  |  |
| Реализация проекта (создание продукта) | 2 |  |  |
| Работа с источниками информации | 2 |  |  |
| Создание текста проектной работы | 2 |  |  |
| Анализ результатов проекта, формулирование выводов | 2 |  |  |
| Создание презентации для выступления | 2 |  |  |
| Создание текста публичного выступления для защиты проекта | 2 |  |  |
| Защита проекта | 2 |  |  |
| Качество оформления текста проектной работы | 2 |  |  |

## Отзыв руководителя

Отзывруководителя на проект/исследование учащегося «\_\_9\_\_» класса

МБОУ г. Иркутска СОШ № 35

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

(ФИО обучающегося)

 по теме

« \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ »

1. Предметная область: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

2. Актуальность и практическая значимость продукта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

3. Уровень самостоятельности при выполнении проекта: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

4. Уровень использования различных видов источников: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

5. Качество оформления работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

6. Положительные стороны работы: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

7. Недостатки и замечания по работе: \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

 (фамилия, имя, отчество руководителя)

Должность руководителя \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Дата \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_