МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ

ОБРАЗОВАНИЕ
НАЦИОНАЛЬНЫЕ ПРОЕКТЫ РОССИИ

СЕТЬ ЦЕНТРОВ ЦИФРОВОГО ОБРАЗОВАНИЯ ДЕТЕЙ «IT-КУБ»

РЕАЛИЗАЦИЯ ОБРАЗОВАТЕЛЬНЫХ ПРОГРАММ ПО ПРЕДМЕТУ «ИНФОРМАТИКА» С ИСПОЛЬЗОВАНИЕМ ОБОРУДОВАНИЯ ЦЕНТРА «ТОЧКА РОСТА»

МОСКВА 2021

С. Г. Григорьев

И. Е. Вострокнутов

М. А. Родионов

И. В. Акимова

О. А. Кочеткова

Реализация образовательных программ по предмету "Информатика" с использованием оборудования центра «Точка роста»

Методическое пособие

под редакцией С. Г. Григорьева

Москва, 2021

Contents

Пояснительная записка

Центры образования цифрового и гуманитарного профилей «Точка роста» были созданы как структурные подразделения общеобразовательных организаций, осуществляющих образовательную деятельность по основным общеобразовательным программам. Данные центры расположены в том числе и в сельской местности и малых городах, направлены на формирование современных компетенций и навыков у обучающихся, в том числе по предметным областям «Технология», «Математика и информатика», «Физическая культура и основы безопасности жизнедеятельности» в 2019 г.

Центры образования естественно-научной направленности «Точка роста» созданы с целью развития у обучающихся естественно-научной, математической, информационной грамотности, формирования критического и креативного мышления, совершенствования навыков естественно-научной направленности, а также для практической отработки учебного материала по учебным предметам «Физика», «Химия», «Биология».

Цель и задачи

- Реализация основных общеобразовательных программ по учебным предметам естественно-научной направленности, в том числе в рамках внеурочной деятельности обучающихся;
- разработка и реализация разноуровневых дополнительных общеобразовательных программ естественно-научной направленности, а также иных программ, в том числе в каникулярный период;
- вовлечение учащихся и педагогических работников в проектную деятельность;
- организация внеучебной деятельности в каникулярный период, разработка и реализация соответствующих образовательных программ, в том числе для лагерей, организованных образовательными организациями в каникулярный период;
- повышение профессионального мастерства педагогических работников центра, реализующих основные и дополнительные общеобразовательные программы.

Создание центра «Точка роста» предполагает развитие образовательной инфраструктуры общеобразовательной организации, в том числе оснащение общеобразовательной организации:

- оборудованием, средствами обучения и воспитания для изучения (в том числе экспериментального) предметов, курсов, дисциплин (модулей) естественно-научной направленности при реализации основных общеобразовательных программ и дополнительных общеобразовательных программ, в том числе для расширения содержания учебных предметов «Физика», «Химия», «Биология»;
- оборудованием, средствами обучения и воспитания для реализации программ дополнительного образования естественно-научной направленности;
- компьютерным и иным оборудованием.

Кроме того, центры «Точки роста» могут выступать в роли пространства для развития цифровой грамотности населения, творческой и проектной деятельности, познавательной активности учащихся, их родителей, педагогов и пр. Данный проект рассчитан на 5 лет.

Целью данного пособия является создание условий для внедрения на уровнях начального общего, основного общего и (или) среднего общего образования новых методов обучения и воспитания, образовательных технологий, обеспечивающих освоение обучающимися основных и дополнительных общеобразовательных программ цифрового, естественно-научного, технического и гуманитарного профилей, обновление содержания и совершенствование методов обучения предметных областей «Технология», «Математика и информатика», «Физическая культура и основы безопасности жизнедеятельности».

Нормативная база

1. Конституция Российской Федерации (принята всенародным голосованием 12.12.1993 с изменениями, одобренными в ходе общероссийского голосования 01.07.2020) — URL: http://www.consultant.ru/document/cons_doc_LAW_28399/(дата обращения: 10.03.2021)

2. Федеральный закон от 29.12.2012 № 273-ФЗ (ред. от 31.07.2020) «Об образовании в Российской Федерации» (с изм. и доп., вступ. в силу с 01.09.2020) — URL: http://www.consultant.ru/document/cons_doc_LAW_140174 (дата обращения: 28.09.2020)

3. Паспорт национального проекта «Образование» (утв. президиумом Совета при Президенте РФ по стратегическому развитию и национальным проектам, протокол от 24.12.2018 № 16) — URL: //https://login.consultant.ru link ?req=doc&base=LAW&n=319308&demo=1 (дата обращения: 10.03.2021)

4. Государственная программа Российской Федерации «Развитие образования» (утверждена Постановлением Правительства РФ от 26.12.2017 № 1642 (ред. от 22.02.2021) «Об утверждении государственной программы Российской Федерации «Развитие образования» — URL: http: //www.consultant.ru document cons_doc_LAW_286474 (дата обращения: 10.03.2021)

5. Стратегия развития воспитания в Российской Федерации на период до 2025 года (утверждена распоряжением Правительства РФ от 29.05.2015 № 996-р «Об утверждении Стратегии развития воспитания в Российской Федерации на период до 2025 года») — URL: http://www.consultant.ru/document/cons_doc_LAW_180402/ — (дата обращения: 10.03.2021)

6. Профессиональный стандарт «Педагог (педагогическая деятельность в дошкольном, начальном общем, основном общем, среднем общем образовании) (воспитатель, учитель)» (ред. от 16.06.2019 г.) (Приказ Министерства труда и социальной защиты РФ от 18 октября 2013 г. № 544н, с изменениями, внесёнными приказом Министерства труда и соцзащиты РФ от 25 декабря 2014 г. № 1115н и от 5 августа 2016 г. № 422н) — URL: // http://профстандартпедагога.рф — (дата обращения: 10.03.2021)

7. Профессиональный стандарт «Педагог дополнительного образования детей и взрослых» (Приказ Министерства труда и социальной защиты РФ от 5 мая 2018 г. № 298н «Об утверждении профессионального стандарта «Педагог дополнительного образования детей и взрослых») — URL: //https://profstandart.rosmintrud.ru/obshchiy-informatsionnyy-blok/natsionalnyy-reestr-professionalnykh-standartov/reestr-professionalnykh-standartov/index.php?ELEMENT_ID=48583 (дата обращения: 10.03.2021)

8. Федеральный государственный образовательный стандарт основного общего образования (утвержден приказом Министерства образования и науки Российской Федерации от 17 декабря 2010 г. № 1897) (ред. 21.12.2020) — URL: https://fgos.ru (дата обращения: 10.03.2021)

9. Федеральный государственный образовательный стандарт среднего общего образования (утверждён приказом Министерства образования и науки Российской Федерации от 17 мая 2012 г. № 413) (ред. 11.12.2020) — URL: https://fgos.ru (дата обращения: 10.03.2021)

10. Методические рекомендации по созданию и функционированию детских технопарков «Кванториум» на базе общеобразовательных организаций (Утверждены распоряжением Министерства просвещения Российской Федерации от 12 января 2021 г. № Р-4) — URL: http://www.consultant.ru/document/cons_doc_LAW_374695/ (дата обращения: 10.03.2021)

11. Методические рекомендации по созданию и функционированию центров цифрового образования «IT-куб» (утверждены распоряжением Министерства просвещения

Российской Федерации от 12 января 2021 г. № Р-5) — URL: http://www.consultant.ru/document/cons_doc_LAW_374572/ (дата обращения: 10.03.2021)

12. Методические рекомендации по созданию и функционированию в общеобразовательных организациях, расположенных в сельской местности и малых городах, центров образования естественно-научной и технологической направленностей («Точка роста») (утверждены распоряжением Министерства просвещения Российской Федерации от 12 января 2021 г. № Р-6) — URL: http://www.consultant.ru/document/cons_doc_LAW_374694/ (дата обращения: 10.03.2021)

Основные понятия и термины

Алгоритм — конечное точное предписание действий, которые необходимо выполнить для решения поставленной задачи.

Ассеты — компоненты, которые представляют собой графику, звуковое сопровождение или скрипты.

Визуализация — метод предоставления абстрактной информации в форме, удобной для зрительного восприятия и анализа явления или числового значения.

Виртуальная реальность (VR) — совокупность технологий, с помощью которых можно создать искусственный мир, физически не существующий, но ощущаемый органами чувств в реальном времени в соответствии с законами физики.

Вспомогательный алгоритм — это алгоритм, выполняющий некоторую законченную часть основного алгоритма. В языке Python может реализовываться в виде функции.

Датчик — средство измерений, предназначенное для выработки сигнала измерительной информации в форме, удобной для передачи, дальнейшего преобразования, обработки и (или) хранения, но не поддающейся непосредственному восприятию наблюдателем. Датчики, выполненные на основе электронной техники, называются электронными датчиками. Отдельно взятый датчик может быть предназначен для измерения (контроля) и преобразования одной физической величины или одновременно нескольких физических величин.

Игровое поле — заранее сконфигурированная площадка с заданиями для робота.

Исполнитель алгоритма — это некоторый объект (техническое устройство, робот, автомат), способный выполнять определённый набор команд алгоритма.

Кортеж — это упорядоченная и неизменяемая последовательность элементов различного типа.

Линейный алгоритм — это алгоритм, в котором команды последовательно выполняются однократно одна за другой.

Оператор — это символ, который выполняет операцию над одним или несколькими операндами.

Оператор цикла — оператор, который выполняет одну и ту же последовательность действий несколько раз, количество повторений либо задано, либо зависит от истинности или ложности некоторого условия.

Переменная — это область памяти компьютера, которая имеет название и хранит внутри себя какие-либо данные.

Скрипт — программа в среде Scratch, которая состоит из блоков-операторов.

Список — это упорядоченная изменяемая последовательность элементов различного типа.

Список (в Scratch) — это сложная переменная, предназначенная для хранения нескольких значений.

Среда Scratch — визуальный язык программирования, позволяющий создавать интерактивные мультимедийные проекты.

Спрайт — один из основных компонентов среды Scratch, для которого пишется программа.

Условный алгоритм — это алгоритм, порядок выполнения команд которого зависит от истинности или ложности некоторого условия.

Условный оператор — оператор, который используется для выбора среди альтернативных операций на основе истинности или ложности некоторого условия.

Циклический алгоритм — это алгоритм, предусматривающий многократное повторение группы команд, называемых телом цикла.

Язык программирования — это набор формальных правил, по которым пишут программы.

Python — это язык программирования, применяемый для разработки самостоятельных программ, а также для создания прикладных сценариев в самых разных областях применения.

Структурирование материалов

Содержание обучения может быть представлено следующими разделами:

- Программирование на Python;
- Методы регистрации данных. Программирование расчётов;
- Технологии кодирования и передачи информации;
- Среда программирования Scratch;
- Среда программирования для Arduino;
- Робототехника;
- Вопросы искусственного интеллекта.

Для каждого раздела подготовлены лабораторные работы с необходимым теоретически материалом, заданиями и указанием к их выполнению. Также имеются дидактические материалы общей направленности, которые можно использовать при подготовке преподавателей и учащихся к занятиям и при выполнении лабораторных работ.

Описание материально-технической базы центра

Для организации работы центров образования цифрового и гуманитарного профилей «Точка роста» рекомендовано следующее оборудование:

- 3D-принтер, тип принтера: FDM, FFF.
- МФУ.
- Ноутбук мобильного класса: производительность процессора (по тесту PassMark — CPU BenchMark http://www.cpubenchmark.net/): не менее 2100 единиц). Объем оперативной памяти: не менее 4 Гб. Объём накопителя SSD/eMMC: не менее 128 Гб.
- Аккумуляторная дрель-винтоверт.
- Многофункциональный инструмент (мультитул).
- Шлем виртуальной реальности: наличие контроллеров: 2 шт. Разрешение: не менее 1440 x 1600 на глаз. Встроенные стереонаушники: наличие. Встроенные микрофоны: наличие. Встроенные камеры: не менее 2 шт. Возможность беспроводного использования.
- Ноутбук виртуальной реальности: разрешение экрана: не менее 1920 x 1080 пикселей. Производительность процессора (по тесту PassMark — CPU BenchMark http://

www.cpubenchmark.net/): не менее 9500 единиц. Производительность графической подсистемы (по тесту PassMark Videocard Bench-mark http://www.videocardbenchmark.net): не менее 11000 единиц. Объём оперативной памяти: не менее 8 Гб. Объём памяти видеокарты: не менее 6 Гб. Объем твердотельного накопителя: не менее 256 Гб.

Фотограмметрическое программное обеспечение.

Квадрокоптер.

Рис. 1. Лаборатория центра

Примерная рабочая программа по предмету «Информатика» с использованием оборудования центра «Точка роста»

На базе центра «Точка роста» обеспечивается реализация образовательных программ технологической направленности, разработанных в соответствии с требованиями законодательства в сфере образования и с учётом рекомендаций Федерального оператора учебного предмета «Информатика».

Образовательная программа позволяет интегрировать реализуемые подходы, структуру и содержание при организации обучения информатики в 5—9 классах, выстроенном на базе любого из доступных учебно-методических комплексов (УМК).

Использование оборудования «Точка роста» при реализации данной ОП позволяет создать условия:

- для расширения содержания школьного образования по информатике;
- для повышения познавательной активности обучающихся в технической области;
- для развития личности ребёнка в процессе обучения информатики, его способностей, формирования и удовлетворения социально значимых интересов и потребностей;
- для работы с одарёнными школьниками, организации их развития в различных областях образовательной, творческой деятельности.

Тематическое планирование

№ п/п	Тема	Содержание	Целевая установка урока	Кол-во часов	Основные виды деятельности обучающихся на уроке/ внеурочном занятии	Использование оборудование
5 класс						
1	Робот. Базовые понятия	История развития робототехники. Введение понятия «робот». Поколения роботов. Классификация роботов. Кибернетическая система. Обратная и прямая связь. Датчики	Вводное занятие. Знакомство. Правила техники безопасности	1	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
2	Знакомство со средой VEXcode VR	Основные фрагменты интерфейса платформы. Панель управления, блоки программы, датчики, игровая площадка, экран датчиков и переменных, кнопки управления	Ознакомить обучающихся с платформой VEXcode VR	1	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
3	Исполнительные механизмы конструкторов VEX	Создание простейших программ (скриптов), сохранение и загрузка проекта	Научить обучающихся создавать простейшие программы (скрипты) на платформе VEXcode VR	2	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
4	Программируемый контроллёр	Математические и логические операторы, блоки вывода информации в окно вывода, блоки трансмиссии	Ознакомить обучающихся с блоками управления роботом (блоки вывода, блоки трансмиссии)	2	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
5	Основные блоки	Блоки управления, блоки переменных, блоки датчиков	Ознакомить обучающихся с группой блоков управления роботом и возможностями программирования с их помощью.	2	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы.	Компьютер, проектор, интерактивная доска

В содержание

6	Датчик местоположения, направление движения	Местоположение VR-робота. Скрипт проекта с датчиком местоположения	Ознакомить обучающихся с **датчиком местоположения**	2	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
7	Датчики цвета	Датчики цвета и их направление. Игровое поле «Дисковый лабиринт»	Ознакомить обучающихся с датчиками цвета (верхний и нижний), движением робота по дисковому лабиринту, рассмотреть отражения данных на панели управления и консоли экрана	2	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
8	Датчик расстояния	Датчик расстояния. Простой лабиринт. Динамический лабиринт	Ознакомить обучающихся с датчиком расстояния, рассмотрение различных типов лабиринта (простой и динамический)	4	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
9	Управление магнитом. Сбор фишек	Блоки группы «Магнит». Игровое поле «Перемещение фишек»	Ознакомить обучающихся с группой «Магнит»	2	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
Итого				18 часов		
6 класс						
10	Знакомство со средой Scratch	Изучение основных элементов интерфейса среды Scratch, приёмы работы со спрайтами, приёмы работы с фоном, составление простых скриптов из различных блоков	Ознакомление со средой Scratch, изучение основных инструментов среды	2	Наблюдение за работой учителя, самостоятельная работа со средой Scratch, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска

В содержание

Продолжение

№ п/п	Тема	Содержание	Целевая установка урока	Кол-во часов	Основные виды деятельности обучающихся на уроке/ внеурочном занятии	Использование оборудование
11	Линейные алгоритмы	Основные приёмы составления линейных алгоритмов в среде Scratch, решение задач на составление линейных алгоритмов	Ознакомление с построением и выполнением линейных алгоритмов, работа с основными блоками в среде Scratch	2	Наблюдение за работой учителя, самостоятельная работа со средой Scratch, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
12	Работа с переменными	Основные приёмы добавления переменных в среде Scratch, использование основных блоков для работы с переменными, основные приёмы составления программ с использованием переменных в среде Scratch	Ознакомление с основами работы с переменными в среде Scratch	2	Наблюдение за работой учителя, самостоятельная работа со средой Scratch, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
13	Условные алгоритмы	Ознакомление с понятием «условный алгоритм», основные приёмы составления условных алгоритмов в среде Scratch, использование основных блоков для составления условных алгоритмов в среде Scratch	Ознакомление с основами работы с условными алгоритмами в среде Scratch	2	Наблюдение за работой учителя, самостоятельная работа со средой Scratch, ответы на контрольные вопросы.	Компьютер, проектор, интерактивная доска
14	Циклические алгоритмы	Ознакомление с понятием «циклический алгоритм», основные приёмы составления циклических алгоритмов в среде Scratch, использование основных блоков для составления циклических алгоритмов в среде Scratch	Ознакомление с основами работы с циклическими алгоритмами в среде Scratch	4	Наблюдение за работой учителя, самостоятельная работа со средой Scratch, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска

В содержание

15	Создание подпрограмм	Ознакомление с возможностью создания подпрограмм в среде Scratch. Раздел Другие блоки, создание блока, параметры блок	Ознакомление с основами работы по созданию блоков-подпрограмм в среде Scratch	2	Наблюдение за работой учителя, самостоятельная работа со средой Scratch, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
16	Блок команд «Управление»	Изучение циклов и ветвлений в среде программирования VEXcode VR	Ознакомить обучающихся с ветвлениями циклами	4	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
17	Проекты «Разрушение замка» и «Динамическое разрушение замка»	Использование датчиков для улучшения алгоритма по очистке территории	Ознакомить обучающихся с выполнением проектов по уборке территории на vr.vex.com	4	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
18	Проект «Детектор линии»	Подсчёт количества линий. Программа для поиска и подсчёта линий	Ознакомить обучающихся с выполнением проектов обнаружению линий	2	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска.
Итого				24 часа		
7 класс						
19	Первые программы на языке Python, основные операторы	Написание простых программ на языке программирования Python, знакомство с операторами присвоения, ввода/вывода данных, разработка программ, реализующих линейные алгоритмы на языке программирования Python	Ознакомление с основами написания программ на языке программирования Python, работа с операторами присвоения, ввода/вывода данных	2	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска

Продолжение

№ п/п	Тема	Содержание	Целевая установка урока	Кол-во часов	Основные виды деятельности обучающихся на уроке/ внеурочном занятии	Использование оборудование
20	Условный оператор if	Формат оператора ветвления if на языке программирования Python, разработка программ, реализующих условные алгоритмы	Ознакомление с условным оператором if на языке программирования Python	2	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
21	Циклы в языке Python	Формат оператора ветвления цикла с предусловием while, оператором цикла с параметром for на языке программирования Python, разработка программ, циклические алгоритмы	Ознакомление с операторами цикла foe, while в языке программирования Python	4	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
22	Списки в языке Python	Понятие «список» в языке программирования Python, создание списка, различные способы задания списка, вывод элементов списка на экран, основные функции по работе со списками в языке программирования Python	Ознакомление с понятием «список» в языке программирования Python	4	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
23	Работа со строками в Python	Понятие «строка» в языке программирования Python, различные способы задания строк, основные функции по работе со строками в языке программирования Python	Ознакомление с понятием «строка» в языке программирования Python	4	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
24	Итоги	Защита индивидуальных или групповых проектов, подведение итогов курса	Защита проекта	2	Самостоятельная индивидуальная или групповая проектная деятельность	Компьютер, проектор, интерактивная доска

В содержание

25	Технологии передачи информации	Понятие информации, свойства информации, технологии передачи информации	Ознакомление с понятием информации, свойства информации, технологии передачи информации	2	Наблюдение за работой учителя, самостоятельная работа, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
26	Кодирование информации	Представление о способах кодирования информации, закрепить умения кодировать информацию	Ознакомление с понятием кодирование, способах кодирования	2	Наблюдение за работой учителя, самостоятельная работа, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
27	Кодирование числовой информации	Определение системы счисления, понятия позиционных и непозиционных системах счисления; основание и алфавит системы счисления; научить переводить числа из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную	Ознакомление с основными понятиями позиционных систем счисления, получения навыков по работе в различных позиционных системах счислениях	2	Наблюдение за работой учителя, самостоятельная работа, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
	Итого			24		
8 класс						
28	Табулирование функций, решение уравнений	Основные приёмы по табулированию функций на языке Python, решение квадратных уравнений на языке Python	Рассмотреть табулирование функции и решение квадратного уравнения на языке программирования Python	2	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
29	Работа с матрицами	Основные способы задания матриц в языке Python, выполнение основных операций с матрицами на языке Python		2	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска

В содержание

Продолжение

№ п/п	Тема	Содержание	Целевая установка урока	Кол-во часов	Основные виды деятельности обучающихся на уроке/ внеурочном занятии	Использование оборудование
30	Физические задачи	Решение физических задач на языке Python, основы физического моделирования с помощью языка программирования		2	Наблюдение за работой учителя, самостоятельная работа со средой программирования Python, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
31	Основные понятия языка программирования Prolog: предикаты, операции над предикатами	Понятие предиката. Операции над предикатами. Правила	Рассмотреть основные понятия языка Prolog, ввести понятие предикат, операции над предикатами: отрицание, конъюнкция, дизъюнкция	2	Наблюдение за работой учителя, самостоятельная работа со средой программирования Prolog, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
32	Встроенные предикаты языка программирования Prolog	Встроенные предикаты для ввода данных, встроенные предикаты для вывода данных, встроенные математические предикаты, встроенные арифметические предикаты	Рассмотреть возможности ввода-вывода данных с помощью встроенных предикатов языка Prolog, построение математических выражений, вычислительных программ	2	Наблюдение за работой учителя, самостоятельная работа со средой программирования Prolog, ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
33	Понятие рекурсивного алгоритма, виды рекурсии. Реализация	Определение рекурсивного правила. Виды рекурсивных правил	Приёмы построения рекурсивных программ, применение различных видов ре	2	Наблюдение за работой учителя, самостоятельная работа со средой программирования	Компьютер, проектор, интерактивная доска

В содержание

	рекурсивных алгоритмов в языке программирования Prolog		курсий для решения задач на языке Prolog		Prolog, ответы на контрольные вопросы	
Итого				12		
9 класс						
34	Знакомство с Arduino. Основные комплектующие	Структура и состав Arduino. История Arduino. Основные электронные компоненты	Вводное занятие. Знакомство с Arduino	2	Слушание объяснений учителя. Наблюдение за работой учителя. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
35	Основы программирования в Tinkercad для Arduino	Обзор датчиков, модулей и исполнительных механизмов. Для разработчика Arduino	Список основного функционала Arduino. Ключевые возможности Tinkercad	2	Слушание объяснений учителя. Наблюдение за работой учителя. Работа в Tinkercad. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
36	Создание первой схемы в TinkerCad	Создание электронной схемы	Познакомится с порядком создания электронных схем	2	Слушание объяснений учителя. Наблюдение за работой учителя. Работа в Tinkercad. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
37	Мигающий светодиод	Сборка и программирование схемы «Мигающий светодиод»	Познакомиться со сборкой и программированием светодиодов	2	Слушание объяснений учителя. Наблюдение за работой учителя. Работа в Tinkercad. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
38	RGB-светодиод	Программирование трёхцветного светодиода	Познакомиться с подключением и программированием RGB-светодиодов	2	Слушание объяснений учителя. Наблюдение за работой учителя. Работа в Tinkercad. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска

В содержание

Продолжение

№ п/п	Тема	Содержание	Целевая установка урока	Кол-во часов	Основные виды деятельности обучающихся на уроке/ внеурочном занятии	Использование оборудование
39	Кнопка — датчик нажатия	Подключение кнопки к Arduino	Познакомиться с подключением и программированием кнопов	2	Слушание объяснений учителя. Наблюдение за работой учителя. Работа в Tinkercad. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
40	Управление сервоприводом	Управление сервоприводом при помощи Arduino	Познакомиться с подключением и программированием сервопривода	2	Слушание объяснений учителя. Наблюдение за работой учителя. Работа в Tinkercad. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
41	Кейс «Светофор»	На основе полученных знаний самостоятельно создаём светофор, отвечающий заданным параметрам		4	Слушание объяснений учителя. Наблюдение за работой учителя Работа в Tinkercad. Ответы на контрольные вопросы	Компьютер, проектор, интерактивная доска
Итого				18		

В содержание

Робототехника

Планируемые результаты освоения учебного предмета с описанием универсальных учебных действий, достигаемых обучающимися

Личностные:

— формирование профессионального самоопределения, ознакомление с миром профессий, связанных с информационными и коммуникационными технологиями;
— формирование умения работать в команде;
— развитие внимательности, настойчивости, целеустремлённости, умения преодолевать трудности;
— формирование навыков анализа и самоанализа.

Предметные:

— формирование понятий о различных компонентах робота и платформы VEXcode VR (программные блоки по разделам, исполнительные устройства, кнопки управления и т. д.);
— формирование основных приёмов составления программ на платформе VEXcode VR;
— формирование алгоритмического и логического стилей мышления;
— формирование понятий об основных конструкциях программирования: условный оператор if/else, цикл while, понятие шага цикла.

Метапредметные:

— освоение способов решения проблем творческого характера в жизненных ситуациях;
— формирование умений ставить цель — создание творческой работы, планирование достижения этой цели, создание вспомогательных эскизов в процессе работы;
— использование средств информационных и коммуникационных технологий для решения коммуникативных, познавательных и творческих задач;
— формирование информационной культуры;
— формирование умения аргументировать свою точку зрения на выбор способов решения поставленной задачи.

Формы контроля

Во время проведения курса предполагается текущий, промежуточный и итоговый контроль.

Текущий контроль проводится на каждом занятии с целью выявления правильности применения теоретических знаний на практике. Текущий контроль может быть реализован посредством следующих форм: наблюдение, индивидуальные беседы, тестирование, творческие работы, проблемные (ситуативные) задачи, практические работы, контрольные вопросы и т. д.

Примеры ситуативных задач по модулю 1

Задача 1. Петя запустил робота, который двигается по следующей программе:

1) стартует с точки A и едет на запад со скоростью $V = 3$ м/мин в течение 60 с;
2) поворачивает на юг и столько же времени движется с удвоенной скоростью $2V$;
3) поворачивает на восток и едет с утроенной скоростью $3V$ такое же время, что на первых двух участках вместе взятых;

4) поворачивает на север и, проехав 6 м за 1,5 мин, добирается до финиша, расположенного в точке *B*.

Вопросы:

1. Какова длина первого участка пути? Ответ дайте в метрах с точностью до целых.
2. С какой постоянной скоростью на всём пути должен двигаться робот, чтобы проехать его за то же время? Ответ укажите в метрах в секунду с точностью до сотых.
3. Найдите расстояние между точкой старта A и точкой финиша B робота. Ответ дайте в метрах с точностью до целых.

Задача 2. Три колёсных робота A1, A2 и A3 одинаковой конструкции должны по очереди пройти лабиринт, двигаясь от входа (синий квадрат) к выходу (зелёный квадрат). Робот A1 содержит в памяти карту лабиринта, на которой отмечены синий и зелёный квадраты и указаны все стенки. Робот A2 не знает карты лабиринта и запрограммирован обходить его по правилу правой руки. Робот A3 не знает карты лабиринта и запрограммирован обходить его по правилу левой руки. Какой из роботов пройдёт лабиринт медленнее всего?

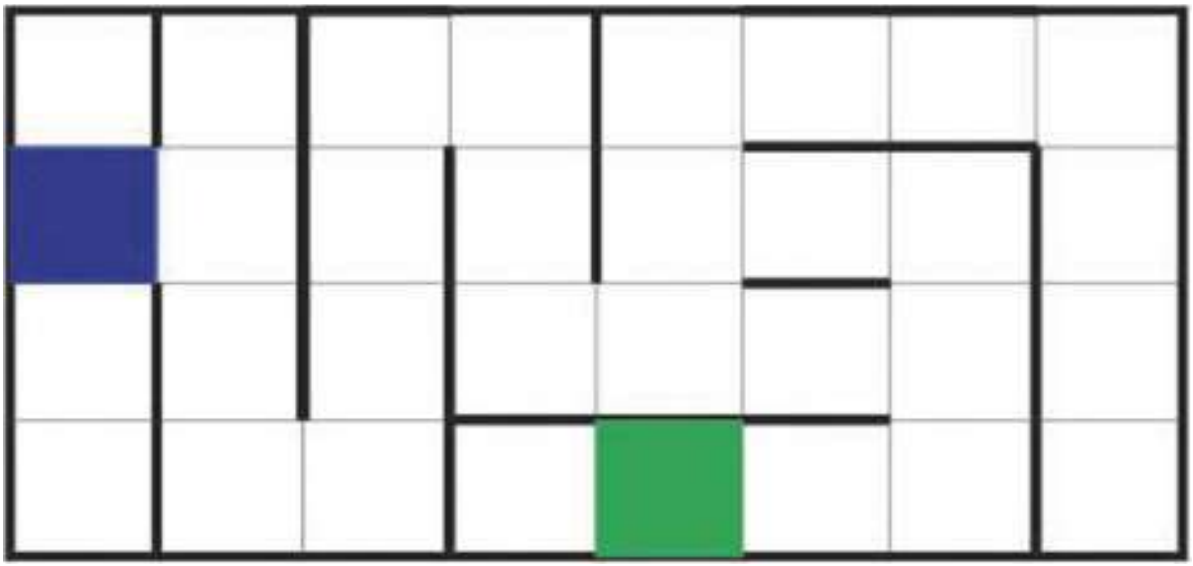

Рис. 2. Вид лабиринта

Промежуточный контроль проводится в рамках промежуточной аттестации после изучения нескольких модулей в виде подготовки и защиты творческих (проектных) работ, соревнований и состязаний.

Пример соревнования «Динамический лабиринт»

Цель: запрограммировать робота на решение лабиринта (прибытие на красный квадрат), в кратчайшие сроки.

Команды состоят только из одного участника.

Правила и подсчёт очков:

1) задача состоит в том, чтобы пройти лабиринт в кратчайшие сроки. Лабиринт считается пройдённым, когда все колёса робота касаются красного квадрата;
2) максимальное время — 180 секунд. Если робот не завершил лабиринт за этот промежуток времени, время будет считаться как 200 секунд;
3) победителем становится команда с лучшим средним временем прохождения лабиринта из двух попыток. Если есть ничья, то в качестве тай-брейка используется лучшее время команды.

Тест по теме «Робот. Базовые понятия»

1. В каком году появилось слово РОБОТ?
А) 1920
Б) 1925
В) 1930
Г) 1935

2. Слово «Робот» — произошёл от чешского слова, которое означает...
А) RoboTech
Б) Robot
В) RobotLand
Г) Robota

3. Кто придумал три закона робототехники?
А) Валли
Б) А. Азимов
В) Г. Галилей
Г) К. Чапек

4. С 1968 г. «столицей роботов» считается
А) Китай
Б) Россия
В) Япония
Г) США

5. Как называется разработанный Aldebaran Robotics человекоподобный робот, поступивший в массовую продажу?
А) T-800
Б) Atlas
В) Pepper
Г) ASIMO

При проведении итоговой аттестации в форме проектной работы задание ориентировано на индивидуальное исполнение. Защита итогового проекта проходит в форме представления обучающимся технического задания на проект, работающего кода, ответов на вопросы преподавателя, обсуждения с учащимися достоинств и недостатков проекта.

Модуль 1. «Платформа VEXcode VR»

В результате изучения данного модуля учащиеся должны:
знать названия различных компонентов робота и платформы:
— контроллёр (специализированный микрокомпьютер);
— исполнительные устройства — мотор, колёса, перо, электромагнит;
— датчики — цвета, расстояния, местоположения, касания;
— панель управления, ракурсы наблюдения робота;
— программные блоки по разделам;
— виды игровых полей (площадок);
— кнопки управления.
— уметь:
— программировать управление роботом;
— использовать датчики для организации обратной связи и управления роботом;
— сохранять и загружать проект.

Урок 1. Робот. Базовые понятия
Урок 2. Знакомство со средой VEXcode VR
Урок 3. Исполнительные механизмы конструкторов VEX
Лабораторная работа 1. Создание простейших программ (скриптов)

Модуль 2. «Программирование робота на платформе»

В результате изучения данного модуля учащиеся должны знать:
— математические и логические операторы;
— блоки вывода информации в окно вывода.
— уметь:
— применять на практике логические и математические операции;
— использовать блоки для работы с окном вывода;
— составлять с помощью блоков математические выражения.
Урок 4. Программируемый контроллёр
Урок 5. Основные блоки
Лабораторные работы 2-3. Программирование блоков управления роботом

Модуль 3. «Датчики и обратная связь»

В результате изучения данного модуля учащиеся должны знать:
— принципы работы датчиков;
— блоки управления датчиками;
— возможности датчиков.
— уметь:
— использовать циклы и ветвления для реализации системы принятия решений;
— решать задачу «Лабиринт».
Урок 6. Датчик местоположения, направление движения
Урок 7. Датчики цвета
Урок 8. Датчик расстояния
Урок 9. Управление магнитом
Лабораторная работа 4. Скрипты с датчиком местоположения
Лабораторная работа 5. Игровое поле «Дисковый лабиринт»
Лабораторные работы 6-8. Простой лабиринт. Динамический лабиринт
Лабораторная работа 9. Игровое поле «Перемещение фишек»

Модуль 4. «Реализация алгоритмов движения робота»

В результате изучения данного модуля учащиеся должны знать:
— условный оператор if/else;
— цикл while;
— понятие шага цикла.
— уметь:
— применять на практике циклы и ветвления;
— использовать циклы и ветвления для решения математических задач;
— использовать циклы для объезда повторяющихся траекторий.
Урок 10. Блок команд «Управление»
Урок 11. Проекты «Разрушение замка» и «Динамическое разрушение замка»
Урок 12. Проект «Детектор линии»
Лабораторная работа 10. Ветвления на базе платформы VEXcode VR
Лабораторная работа 11. Циклы на базе платформы VEXcode VR
Лабораторная работа 12. Блок «Всегда», блок «Прерывания» и блок «Ждать пока»
Лабораторные работы 13—15. Проект по уборке территории
Лабораторная работа 16. Поиск и подсчёт линий

Модуль 5. «Творческий проект»

При выполнении творческих проектных заданий школьники будут разрабатывать свои собственные программы.

Планы учебных занятий

Урок 2. Знакомство со средой VEXcode VR

Тип урока: комбинированный.

Цель урока: знакомство с интерфейсом VEXcode VR.

Планируемые результаты

Предметные: получение навыков работы с панелью управления, получение навыков по созданию первых программ в среде VEXcode VR.

Метапредметные: умение самостоятельно определять цели своего обучения, ставить и формулировать для себя новые задачи в учёбе и познавательной деятельности; умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; сформированность их мотивации к обучению и целенаправленной познавательной деятельности.

Время реализации: 1 академический час.

Оборудование и материалы: компьютеры с выходом в сеть Интернет, проектор, интерактивная доска.

Ход урока

1. Этап постановки цели и задач урока, мотивации к учебной деятельности — 15 минут.

Деятельность учителя — подача нового материала с демонстрацией платформы http://vr.vex.com.

VEXcode VR облегчает изучение информатики и робототехники, позволяя обучающимся кодировать виртуального робота, находясь в любом месте, программируя в среде блочного кодирования. VEXcode VR основан на VEXcode, том же интерфейсе программирования, который используется для робототехнических платформ VEX 123, GO, IQ и V5.

Показать удобство VEXcode VR для различных образовательных учреждений.

VEXcode VR — это способ обогатить опыт компьютерных наук для обучающихся после того, как они открыли для себя программирование в среде Scratch и азарт образовательной робототехники с различными аппаратными платформами. VEXcode VR позволяет учащимся продолжать заниматься робототехникой, даже если поблизости нет физического робота.

Кроме того, VEXcode VR содержит функции, призванные помочь в освоении программирования и сделать STEM и информатику доступными для большего числа обучающихся. Эти функции включают простоту использования, немедленную обратную связь, возможность сделать обучение видимым и помочь с различными реализациями в классе.

Легко использовать

VEXcode VR на 100% основан на веб-технологиях, поэтому запустить VEXcode VR просто. Пользовательский интерфейс (ПИ) упрощает навигацию — команды разделены на категории, ПИ построен, избегая большого списка команд в меню. Команды также имеют цветовую кодировку, поэтому пользователи могут легко находить связанные блоки. Область программирования всегда видна; приглашение учащихся начать программирование. VEXcode VR использует готовых роботов и команды трансмиссии. Это позволяет пользователям управлять своим виртуальным роботом за считаные секунды.

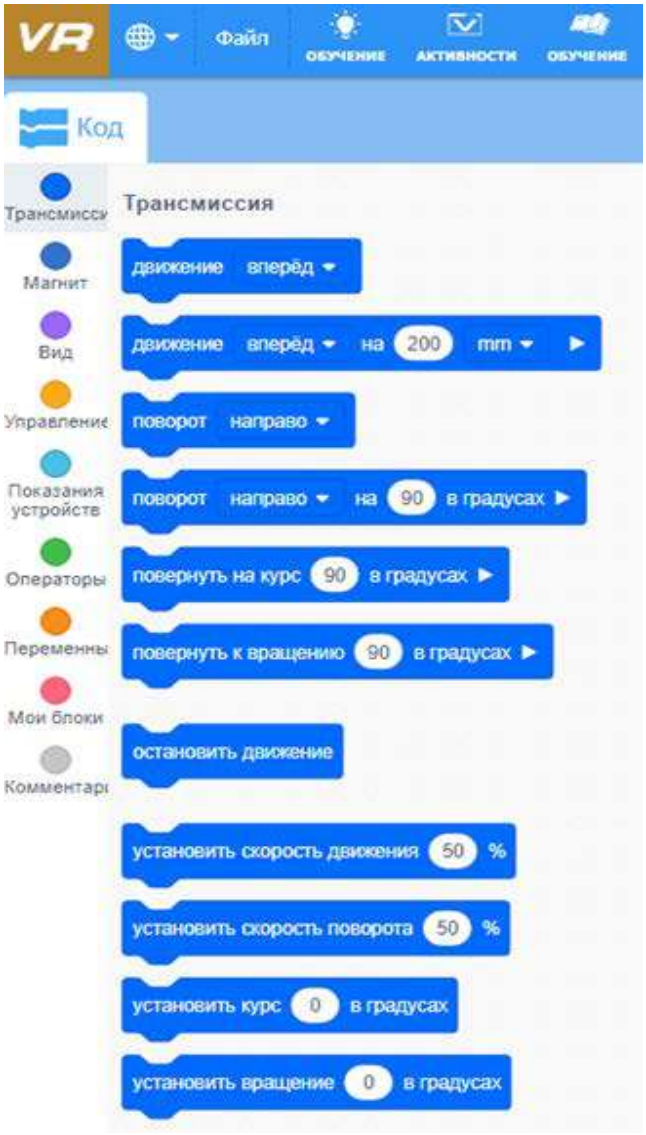

Рис. 3. Блок команд «Трансмиссия»

Робот VEX VR оснащён датчиками, элементами управления и множеством физических функций. В VEXcode VR есть только один робот, и он уже настроен заранее. Это устраняет необходимость в настройке конфигурации робота или заранее определённом шаблоне проекта.

Обратная связь

VEXcode VR стимулирует проведение экспериментов и игр. Когда обучающиеся запускают проект, они могут сразу увидеть, дал ли их робот желаемый результат. Учителя легко контролируют успехи обучающихся. Проект в VEXcode VR всегда будет работать одинаково — это не всегда происходит с физическим роботом. Это позволяет учителям и ученикам сосредоточиться на логике программирования, а не на физике робота или на игровом поле, на котором работает робот.

Игровое поле (площадка) VEXcode VR

Робот VR всегда начинает выполнение программы с одной и той же точки. Обучающиеся могут добавлять блоки во время работы своего проекта, останавливать проект в любой момент и перезагружать свою виртуальную площадку одним щелчком мыши. Блоки, не подключённые к основному скрипту, просто игнорируются при запуске проекта.

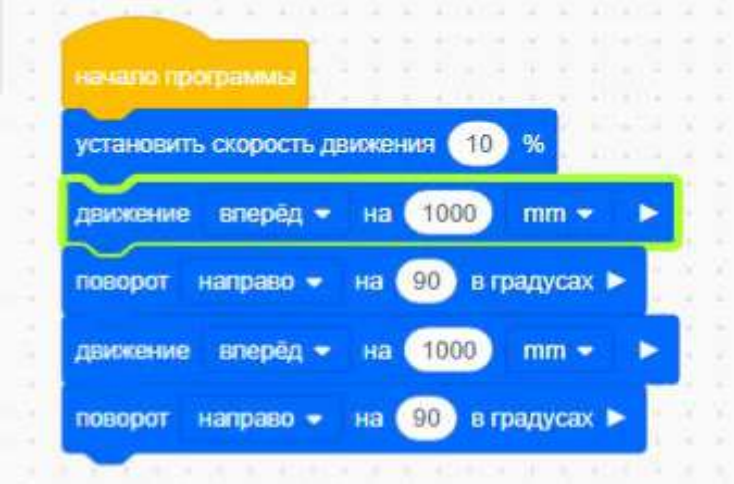

Рис. 4

В VEXcode VR ошибок нет. Обучающиеся могут делать логические ошибки при кодировании, но они не будут разочарованы тем, что их проекты не компилируются и не запускаются. Способность VEXcode VR обеспечивать немедленную обратную связь и простота использования побуждают обучающихся учиться в процессе написания кода создавать код с помощью небольших фрагментов, постепенно формируя окончательный вариант программы (скрипта).

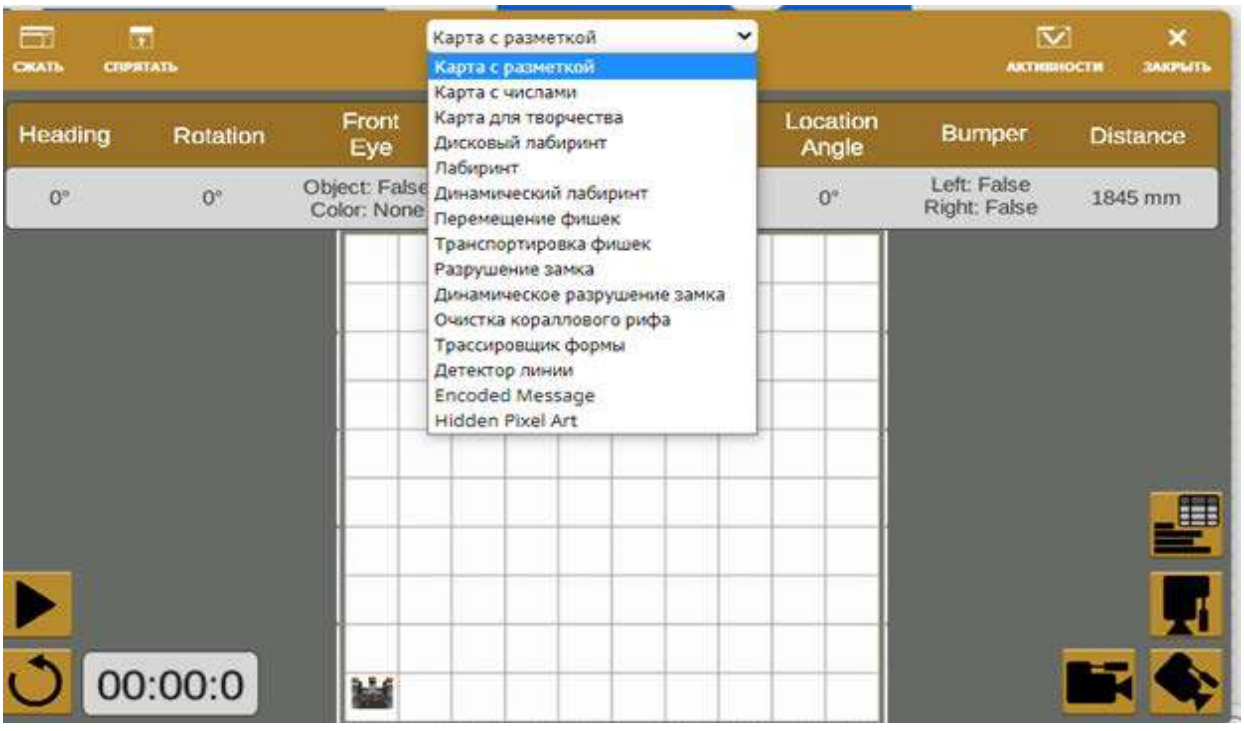

Рис. 5. Игровое поле с выпадающим меню выбора игрового поля

Обучение стало наглядным

Окно «Игровое поле» в VEXcode VR содержит приборную информационную панель, на которой отображаются все данные датчиков от VR-робота. Каждый раз, когда работает робот VR, учащиеся могут видеть обновление данных датчика в режиме реального времени, получая информацию о том, как эти данные могут быть использованы.

Рис. 6. Приборная панель с показаниями датчиков

VEXcode VR также выделяет блоки в проекте, когда эти блоки выполняются зелёным контуром. Эта функция позволяет учащимся наблюдать за ходом выполнения своих проектов. Когда проект VEXcode VR запущен, выполняемый блок окружён светящейся зелёной рамкой. Эта обратная связь помогает обучающимся понять, почему робот VR выполняет определённый манёвр. Эти данные в реальном времени вместе с выделением выполненных блоков могут помочь обучающимся понять, как робот принимает решение, сделать этот процесс более заметным и конкретным.

VEXcode VR также помогает организовать дистанционную работу. Во всех этих (и не только) ситуациях VEXcode VR может стать отличным дополнением для физических роботов.

Рис. 7. Конфигурация датчиков робота

Управление роботом

Робот VR имеет следующие элементы управления:

- трансмиссия с гироскопом. Это включает категорию блоков «Трансмиссия» в наборе инструментов VEXcode VR;
- функция «рисования пером», позволяющая размещать перо вверх (чтобы не рисовать) или вниз (для рисования);
- электромагнит для поднятия дисков с металлическими сердечниками.

Физические характеристики робота

Робот VR обладает следующими физическими характеристиками:

- колёса имеют диаметр 50 мм;

- колёсная база (расстояние между центром переднего колеса и центром заднего колеса) составляет примерно 50,8 мм;
- Длина робота VR составляет 133 мм.

Датчики роботов

Робот VR оснащён следующими датчиками:

- моторные энкодеры с углом поворота 360 ° на оборот колеса;
- передний датчик глаза также действует как датчик расстояния и возвращает расстояние до обнаруженного объекта в миллиметрах и дюймах;
- гироскопический датчик, встроенный в трансмиссию. По часовой стрелке положительный.

Вид сверху на робота VEXcode VR

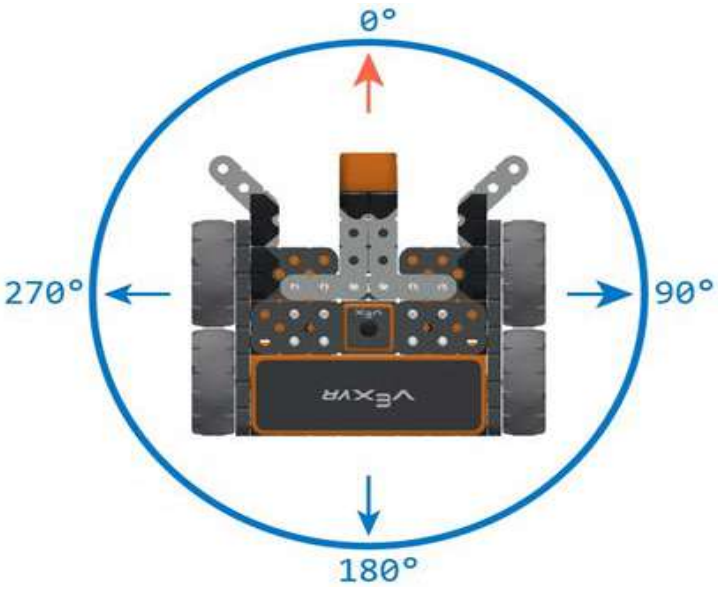

Рис. 8. Направления курса, задаваемые блоком «Курс»

Два датчика цвета, один обращён вперёд, а другой вниз. Эти датчики могут определять наличие объекта. Если есть, датчик также может определять цвет (красный, зелёный, синий, нет).

Датчик местоположения, который считывает координаты (X, *Y*) от центральной поворотной точки VR-робота.

2. Этап актуализации знаний и пробного учебного действия — 15 минут.

Учитель предлагает обучающимся под его контролем выполнить некоторые действия на платформе:

- запрограммировать робота на движение вперёд и назад. Блок программ «Трансмиссия»;
- запрограммировать робота на движение на определенное расстояние. Блок программ «Трансмиссия»;
- запрограммировать робота на повороты в градусах и по курсу. Блок программ «Трансмиссия»;
- проверить загрузку различных учебных полей.

3. Этап повторения нового материала — 5 минут.

Учитель повторяет кратко новый материал: роботы, датчики, скрипты, игровые поля, как заставить робота двигаться, поворачивать.

4. Этап проверки понимания и первичного закрепления — 5 минут.
Учитель задаёт вопросы ученикам:
Для чего нужна программа?
Что такое датчики и для чего они нужны?

5. Информация о домашнем задании, инструктаж по его выполнению — 3 минуты.
Задание 1. Зайти на платформу VEX code VR.
Задание 2. Написать программу движения робота вперёд и назад на 1000 мм (одна клетка поля равна 200 мм).

6. Этап рефлексии деятельности на уроке — 2 минуты.
Учитель интересуется у обучающихся их впечатлениями от урока, что им понравилось и что было непонятно.

Урок 3. Исполнительные механизмы конструкторов VEX

Тип урока: комбинированный.
Цель урока: создание простейших программ (скриптов), сохранение и загрузка проекта.

Планируемые результаты

Предметные: получение навыков написания программ (скриптов) на платформе VEXcode VR.

Метапредметные: умение соотносить свои действия с планируемыми результатами; осуществлять контроль своей деятельности в процессе достижения результата; определять способы действий в рамках предложенных условий и требований, корректировать свои действия в соответствии с изменяющейся ситуацией.

Личностные: стимулирование поиска вариантов на основе имеющихся знаний; готовность и способность обучающихся к саморазвитию и личностному самоопределению; развитие находчивости, умения преодолевать трудности для достижения намеченной цели.

Время реализации: 1 академический час.

Оборудование и материалы: компьютеры с выходом в сеть Интернет, проектор, интерактивная доска.

Ход урока

1. Этап постановки цели и задач урока, мотивации к учебной деятельности 15 — минут.
Деятельность учителя — подача нового материала с демонстрацией платформы http://vr.vex.com.
Примеры различных программ, созданных с помощью платформы, можно открыть в меню **Файл-Открыть примеры.**

Рис. 9. Примеры программ, созданных на платформе VEXcode VR

Создайте простую программу движения робота по периметру игрового поля **Карта с разметкой** (размеры клетки 200 × 200 мм).

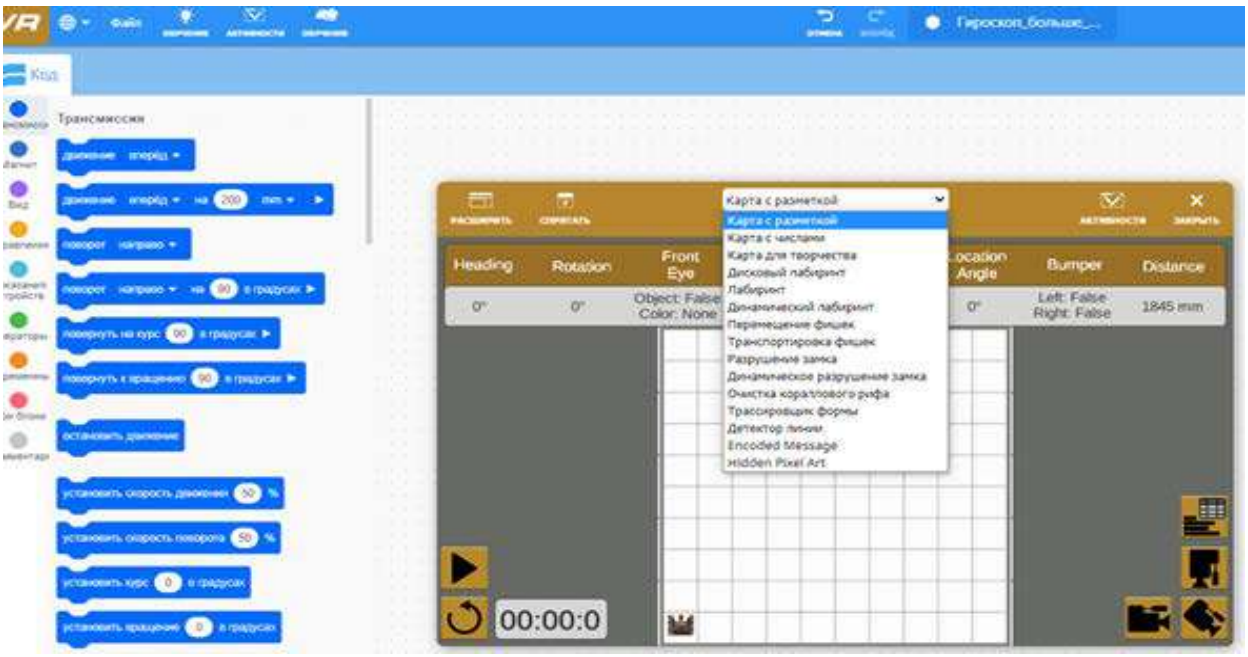

Рис. 10. Игровое поле **Карта с разметкой**

Простая программа будет выглядеть следующим образом:

Рис. 11. Скрипт обхода игрового поля по периметру

Можно немного усложнить программу. Пусть робот при обходе поля рисует линию. Для этого перейдём в блок программ **Вид** и перед началом движения поставим два блока, показанные на рисунке ниже.

Рис. 12. Скрипт рисования роботом квадрата

Созданную программу необходимо сохранить на своём компьютере или планшете. Для этого нужно кликнуть мышкой на верхней панели и ввести имя файла.

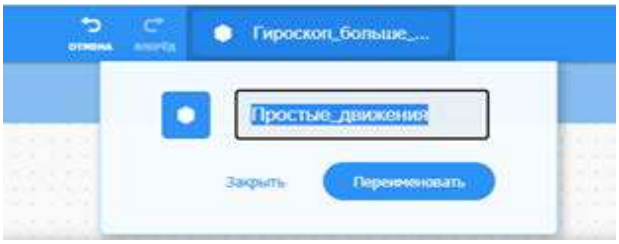

Рис. 13. Переименование рабочего файла

Сохранить этот файл на своё устройство можно через меню **Файл-Сохранить на ваше устройство**, файл будет скачан в папку **Загрузки** рабочего устройства.

Также рассмотрим позиционирование робота на игровой площадке с помощью датчика положения (гироскопа).

Игровая площадка разделена на координатные плоскости (декартовы координаты) как показано на рисунке 14.

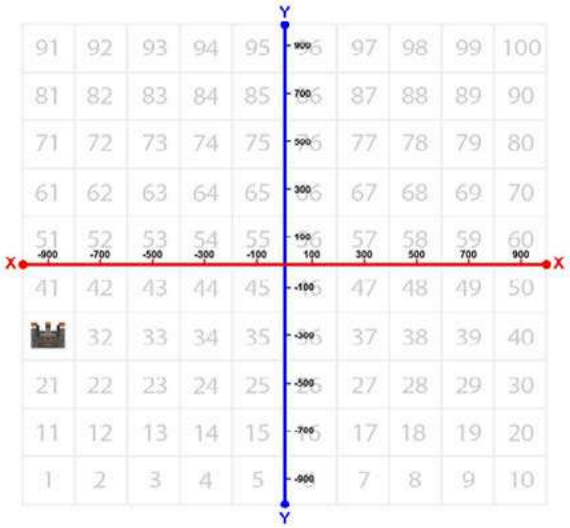

Рис. 14. Координатная плоскость игрового поля

Движение по координатной плоскости задаётся следующим образом:

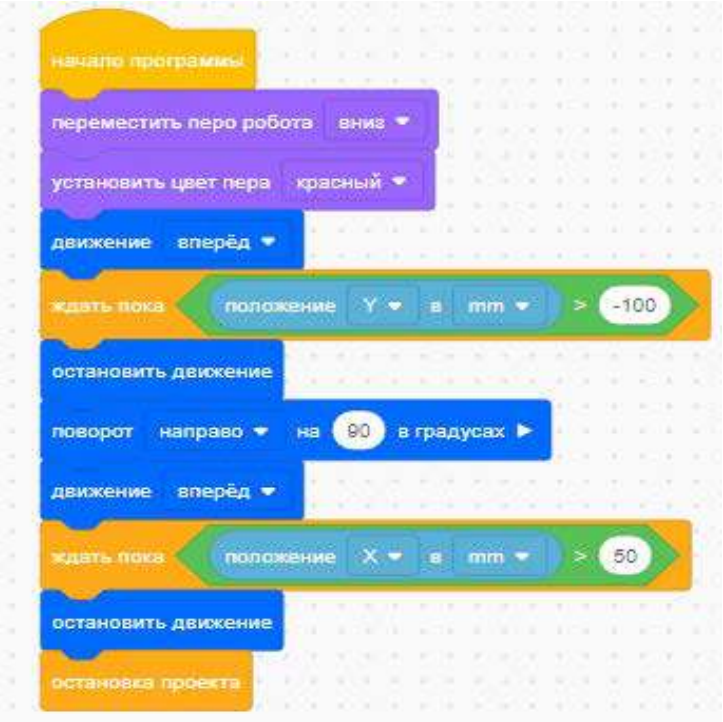

Рис. 15. Перемещение робота по координатам

2. Этап актуализации знаний и пробного учебного действия — 15 минут.

Учитель предлагает обучающимся под его контролем выполнить некоторые действия на платформе, используя знания, полученные в ходе изложения нового материала.

3. Этап актуализации знаний и пробного учебного действия — 15 минут.

Учитель предлагает обучающимся под его контролем выполнить некоторые действия на платформе:

Задание 1. Напишите скрипт рисования роботом квадрата со сторонами разного цвета.

Задание 2. Напишите скрипт рисования роботом ромба со сторонами разного цвета, используя гироскоп.

4. Этап повторения нового материала — 5 минут.

Учитель повторяет кратко новый материал: программирование простых движений робота, датчики, скрипты, игровые поля, как заставить робота двигаться, поворачивать, рисовать.

5. Этап проверки понимания и первичного закрепления — 5 минут.

Учитель задаёт вопросы ученикам:

- Для чего нужна программа «Трансмиссия»?
- Какой блок программ отвечает за движение робота?
- Как регулировать скорость движения робота?
- Как работает блок программ «Курс»?
- Как направить движение робота по диагонали игрового поля?
- Как управлять магнитом и для чего?

6. Информация о домашнем задании, инструктаж по его выполнению — 3 минут.

Задание. Написать программу рисования на игровом поле роботом инициалов обучающегося.

7. Этап рефлексии деятельности на уроке — 2 минут.

Учитель интересуется у обучающихся об их впечатлениях от урока, что им понравилось и что было непонятно.

Урок 10. Блок команд «Управление»

Цель урока: ознакомить обучающихся с ветвлениями и циклами.

Тип урока: комбинированный.

Планируемые результаты

Предметные: получение навыков по использованию условного оператора и операторов цикла, разработка программ, реализующих разветвляющийся и циклический алгоритмы.

Метапредметные: делать выводы в процессе работы и по её окончании, корректировать намеченный план, ставить новые цели; умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, корректировать свои действия в соответствии с изменяющейся ситуацией; умение оценивать правильность выполнения учебной задачи.

Личностные: формирование мотивации к обучению и целенаправленной познавательной деятельности; формирование целеустремлённости и усидчивости в процессе творческой, исследовательской работы и учебной деятельности.

Время реализации: 1 академический час.

Оборудование и материалы: компьютеры с выходом в сеть Интернет, проектор, интерактивная доска

Ход урока

2. Этап постановки цели и задач урока, мотивации к учебной деятельности — 15 минут.

Деятельность учителя: подача нового материала с демонстрацией командам управления в среде vr.vex.com.

Блок команд управления в VEX VR

Вы уже знакомы с понятиями циклов и ветвлений, поэтому самое время перейти к изучению этих структур в среде программирования VEX VR.

Почему данная группа блоков имеет название «Управление»? На самом деле всё довольно просто. Эти блоки позволяют управлять поведением нашего робота в зависимости от обстановки и окружающих его факторов. Без этих блоков нельзя получить интеллектуальную систему, способную ориентироваться в пространстве без дополнительной помощи со стороны оператора или команды разработчиков. На рисунке 16 показан блок команд управления. Обратите внимание, что многие блоки схожи между с собой по названиям и, соответственно, обладают похожими свойствами. Сегодня познакомимся с блоками **Ждать** и **Повторять** (рис. 16).

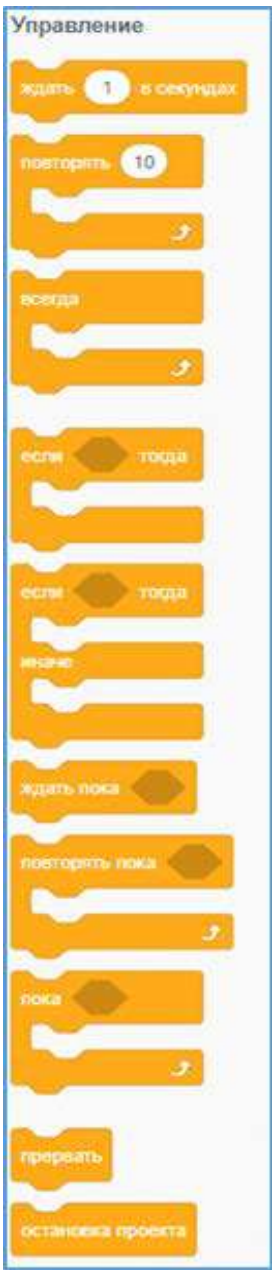

Рис. 16. Блок команд управления

Рассмотрим блок **Ожидания** (рис. 17).

Рис. 17. Блок **Ожидания**

Данный блок позволяет удерживать действия, которые происходили до его вызова. Рассмотрим пример (рис. 18).

Рис. 18. Пример программы с использованием блока **Ожидания**

Что произойдёт в данном отрывке кода? Робот будет двигаться вперёд, после чего остановится. Если же убрать данный блок, то, соответственно, визуально ничего не произойдёт, так как робот мгновенно выполнит обе команды и даже не сдвинется с места.

Важно! Блоки программы после блока **Ожидания** не будут выполняться, пока не истечёт время, установленное внутри этого блока.

Перейдём к блоку **Повторять**. На самом деле, как вы уже поняли, данный блок повторяет некоторое число действий, количество которых определено внутри этого блока. Рассмотрим программу для изображения квадрата (рис. 19).

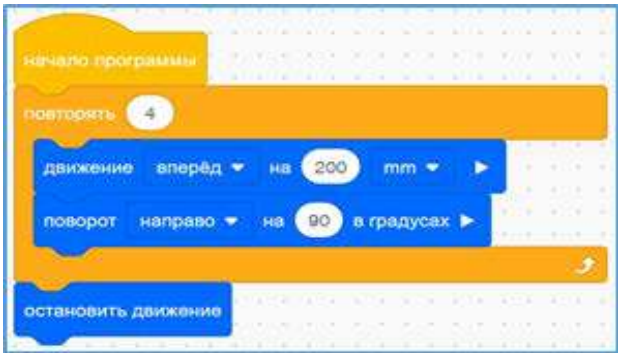

Рис. 19. Программа по объезду квадрата

Давайте более подробно рассмотрим данную программу. Почему бы просто не использовать группу из одинаковых блоков для решения задачи? Никто не запрещает использовать несколько одинаковых блоков вместо циклов, но при написании кода большего объёма получим гораздо больше строк, что, несомненно, усложнит читаемость всей программы. Для сравнения на рисунке 20 показан тот же самый код, но без использования цикла.

Рис. 20. Объезд по квадрату без цикла

Как вы видите, циклы упрощают читаемость кода, не влияя на его функциональность.

3. Этап актуализации знаний и пробного учебного действия — 15 минут.

Учитель предлагает обучающимся под его контролем выполнить некоторые действия на платформе:

Задание. Написать программу для объезда шестиугольника, при этом используя только блоки движения без указания расстояния, блок поворота без указания угла и блок ожидания.

4. Этап повторения нового материала — 5 минут.

Учитель повторяет кратко новый материал: блоки **Управления**, блок **Ожидания** и блок **Повторения**.

5. Этап проверки понимания и первичного закрепления — 5 минут.

Учитель задаёт вопросы ученикам:

- Почему блоки управления так называются?
- Зачем нужны блоки управления?
- Что позволяет делать блок ожидания?
- Что делает блок повторения?

6. Информация о домашнем задании, инструктаж по его выполнению — 3 минут.

Задание 1. Реализовать алгоритм объезда роботом треугольника и прямоугольника.

Задание 2. Реализовать алгоритм объезда роботом треугольника и прямоугольника без использования блоков поворота с указанием угла поворота (необходимо использовать только блок ожидания).

7. Этап рефлексии деятельности на уроке — 2 минуты.

Учитель интересуется у обучающихся об их впечатлениях от урока, что им понравилось и что было непонятно.

Лабораторная работа 10
Ветвления на базе платформы VEXcode VR

Теоретическая часть

Иногда роботу бывает необходимо единожды принять выбор относительно своего текущего действия. Это могут быть повороты направо или налево, отъезды вперёд или назад в определённый момент времени без необходимости перепроверки действий, как в циклах. В этом случае лучше всего использовать блоки **Если-тогда** и **Если-тогда-иначе**, которые позволяют преобразовать действия, совершаемые роботом, в систему с выходами да/нет.

Практическая часть

Цель работы: ознакомить обучающихся с ветвлениями на базе платформы vr.vex.com.

Ход лабораторной работы

На предыдущем занятии рассмотрели блоки ожидания и повторения и сегодня продолжим изучение остальных блоков модуля управления.

В первую очередь обратим внимание на блок **Если-тогда** (рис. 21).

Рис. 21. Блок **Если-тогда**

Тело данного блока выполняется в том случае, если условие, обозначенное в шестиугольной вставке, является правдой, и не выполняется в обратном случае. Рассмотрим следующий пример (рис. 22).

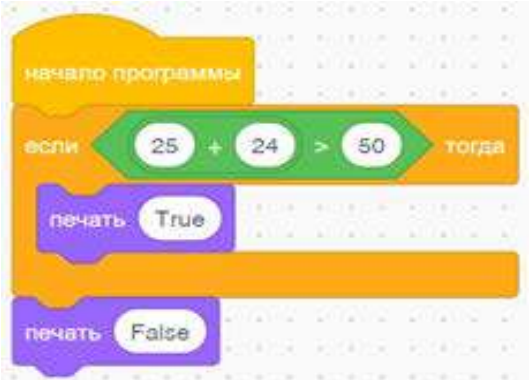

Рис. 22. Пример использования условного блока

Давайте подробнее рассмотрим, что будет происходить в данном примере. Как вы видите, в качестве условия в блоке дано логическое выражение, содержащее справа сумму чисел, а слева константу. Если сравнение окажется правдой, то в окно вывода будет напечатано True, если условие окажется ложным — False.

Кроме того, существует ещё один блок управления, схожий с вышеуказанным, но при этом имеющий ещё одну дополнительную ветвь иначе (рис. 23).

Рис. 23. Блок **Если-тогда-иначе**

Рассмотрим следующую программу для понимания разницы между блоками **Если-тогда** и **Если-тогда-иначе** (рис. 24).

Рис. 24. Демонстрация работы **Если-тогда-иначе**

Рассмотрим код данной программы. Выражение *после если* в данном случае является ложным, поэтому будет выполняться часть после *иначе*, то есть перо робота станет зелёного цвета и на экране появится надпись Second Over. Обратите внимание, что Over будет выведено в любом случае, так как находится за пределами блока условного оператора.

Задание 1. Написать программу для вывода на экран True или False в зависимости от решения простых математических выражений, которые учащиеся должны разместить внутри условия условного блока.

Задание 2. Реализовать программу для поворота роботом перед препятствием на произвольный угол.

Контрольные вопросы:

1. Как можно использовать блок **Если-тогда**?
2. В чём отличие блоков **Если-тогда** и **Если-тогда-иначе**?
3. В каких ситуациях лучше всего использовать блок **Если-тогда-иначе**?

Лабораторная работа 11
Циклы на базе платформы VEXcode VR

Теоретическая часть

Известно, что бывают ситуации, при которых необходимо проверять условие несколько раз, прежде чем принять окончательное решение. Можно использовать совокупность блоков повторения и условные блоки, но для корректной работы повторений должно быть огромное число (десятки миллионов) и подбор точного числа может сильно затруднить общее время создания программы. Во избежание этого в VEX VR используются циклы **Повторять пока** и **Пока.**

Практическая часть

Цель работы: ознакомить обучающихся с циклами на базе платформы vr.vex.com

Ход лабораторной работы

В VEX VR используются циклы **Повторять пока** и **Пока** (рис. 25).

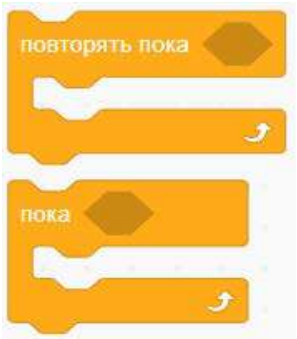

Рис. 25 — Блоки циклов

Рассмотрим простой пример программы, который покажет отличия между двумя видами циклов (рис. 26).

Рис. 26. Отличия циклов в VEX VR

На первый взгляд может показаться, что отличия в условиях блоков циклов должны привести к выполнению абсолютно разных действий, но это не так. Природа блока **Повторять пока** такова, что действия внутри блока будут выполняться до тех пор, пока условие будет ложным. Об этом важно помнить при составлении программы.

Давайте пока отложим блок **Повторять пока** и напишем простую программу для вывода на экран нескольких строк текста в зависимости от показаний датчика расстояния (рис. 27).

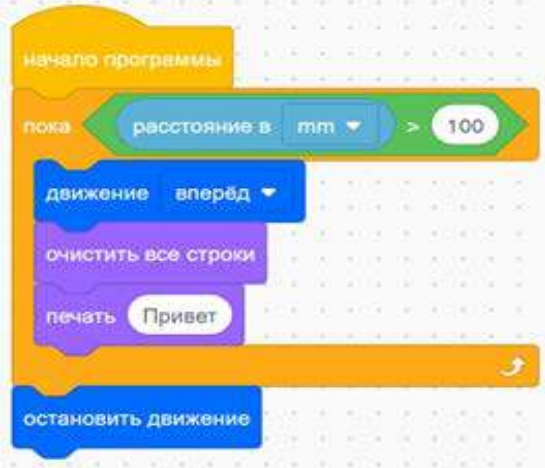

Рис. 27. Работа с циклом **Пока**

Давайте построчно рассмотрим данную программу. В первую очередь создаём блок цикла, который будет работать до тех пор, пока робот не увидит перед собой какое-либо препятствие. На пути всего следования до препятствия робот будет выводить в консоль вывода слово «Привет» огромное число раз в силу высокой производительности управляющего контроллера. Если добавить к программе всего одну строку, то слово «Привет» будет выведено только один раз (рис. 28).

Рис. 28. Вывод одного слова на экран во время использования цикла

Задание 1. Написать программу для вывода на экран *привет*. При каждом проходе цикла привет должно выводиться с новой строки.

Задание 2. Написать программу по изображению роботом квадрата (в качестве повторения пройденного материала).

Контрольные вопросы:

1. В чём отличия блоков **пока** и **повторять пока**?
2. Приведите примеры программы, использующей блок **пока** (в виде псевдокода).
3. Приведите примеры программы, использующей блок **повторять пока** (в виде псевдокода).

Лабораторная работа 12
Блок Всегда, блок Прерывания и блок Ждать пока

Теоретическая часть

На предыдущем занятии рассмотрели блоки циклов и сегодня продолжим изучение оставшихся блоков модуля управления: блока **Всегда**, блока **Прерывания** и блока **Ждать пока**.

Блок **Всегда** представляет собой бесконечный цикл без условия, который будет выполняться на протяжении всей программы независимо от действий, происходящих внутри и снаружи него

Практическая часть

Цель работы: ознакомить обучающихся с блоками **Всегда**, блоками **Прерывания** и блоками **Ждать пока.**

Ход лабораторной работы

Рассмотрим две программы, которые являются абсолютно идентичными, но при этом продемонстрируют принцип работы блока **Всегда** (рис. 29).

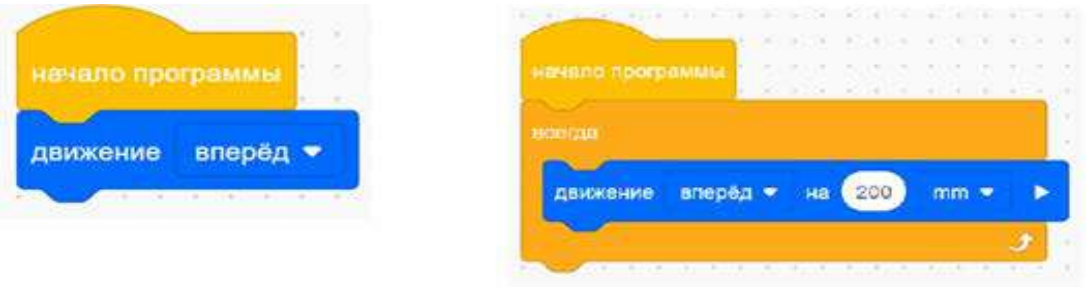

Рис. 29. Аналог бесконечного цикла

Как вы можете заметить, в данной программе блок движения включает себя бесконечный цикл, что позволяет роботу двигаться бесконечно вперёд при добавлении этого блока в программу.

Для того чтобы прервать выполнение бесконечного цикла, можно использовать блок **Прерывания**.

Рассмотрим программу, в которой робот будет ехать вперёд до тех пор, пока датчик расстояния не увидит перед собой некоторое препятствие на расстоянии 100 мм (рис. 30).

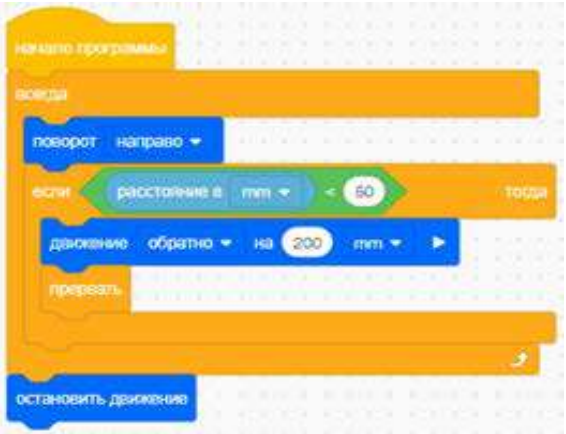

Рис. 30. Программа для демонстрации работы цикла прерывания

Данная программа выполняет все те же действия, что и использование цикла Пока в предыдущем уроке, но при этом наглядно демонстрирует способ прерывания цикла Всегда. Использование блока **Прервать** помогает избежать поломок робота, так как обычно используется в контексте аварийного прерывания выполнения программы. Давайте рассмотрим более сложный вариант программы, в котором использование цикла будет оправдано срабатыванием датчика расстояния на расстоянии меньшем, чем планируемое (рис. 31).

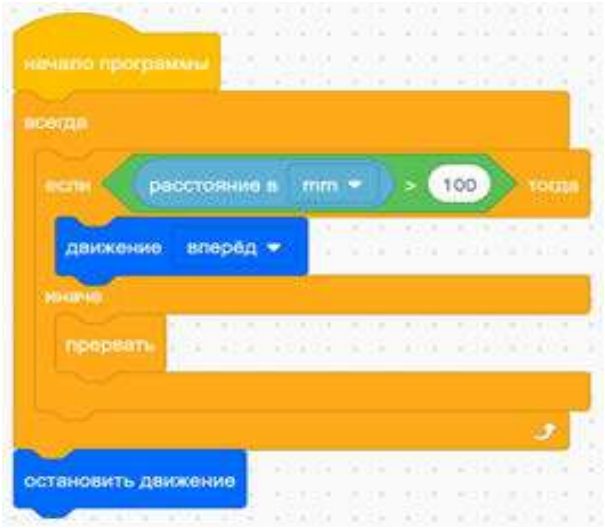

Рис. 31. Пример программы

В данной программе робот будет совершать поворот до тех пор, пока не обнаружит перед собой препятствие. После этого он отъедет назад на 200 мм и остановит своё движение. Данный алгоритм может быть применим к случаям, когда на роботе установлено техническое зрение и ему необходим лучший обзор для идентификации перед собой объектов.

Кроме того, необходимо обратить внимание на блок **Ждать пока** (рис. 32), который позволяет удержать выполнение действий до этого блока.

Рассмотрим программу, которая может помочь в прохождении некоторых испытаний в среде VEX VR.

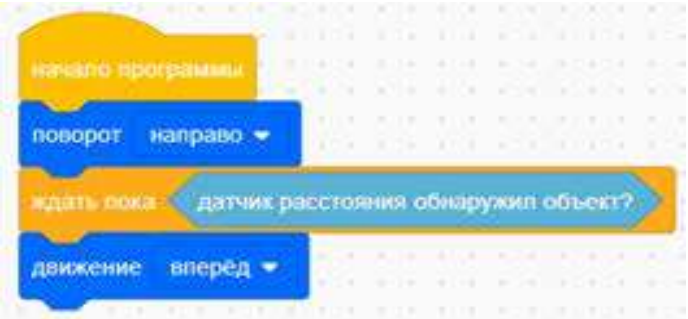

Рис. 32. Блок **Ждать пока**

В данной программе робот при обнаружении объекта постарается вытолкнуть его с поля. Данный код поможет в решении задач на последующих уроках.

Задание 1. Написать программу объезда всего периметра поля «Карта с разметкой» с использованием блока **Ждать пока.**

Задание 2. Написать программу по завершении роботом действий при обнаружении под собой красного цвета.

Задание 3. Реализовать программу для активации роботом отъезда назад при первоначальном движении робота вперёд.

Контрольные вопросы:

1. Зачем нужен блок **Прерывания**?
2. Приведите примеры программы, использующей блок **Прерывания** (в виде псевдокода)
3. Как работает блок **Ждать пока**?

Дидактические материалы

1. Платформа программирования роботов VEXCode VR [Электронный ресурс] // URL: https://vr.vex.com (Дата обращения: 15.04.2021).
2. Информатика. Уровень1-Блоки [Электронный ресурс] //URL: https://education.vex.com/stemlabs/cs/computer-science-level-1-blocks (Дата обращения: 15.04.2021).
3. Официальный сайт среды программирования Scratch [Электронный ресурс] // URL: https://scratch.mit.edu/ (Дата обращения: 15.04.2021).
4. STEM Education channel by Mark Johnston // URL: https://www.mjstem.com/ (Дата обращения: 15.04.2021).

Программирование на Python

Планируемые результаты освоения учебного предмета с описанием универсальных учебных действий, достигаемых обучающимися

Личностные:

— формирование умения самостоятельной деятельности;
— формирование умения работать в команде;
— формирование коммуникативных навыков;
— формирование навыков анализа и самоанализа;
— формирование целеустремлённости и усидчивости в процессе творческой, исследовательской работы и учебной деятельности.

Предметные:

— формирование понятий «алгоритм», «программа»;
— формирование понятий об основных конструкциях языка программирования Python: оператор ветвления if, операторы цикла while, for, вспомогательных алгоритмов;
— формирование понятий о структурах данных языка программирования Python;
— формирование основных приёмов составления программ в программировании на языке программирования Python;
— формирование алгоритмического и логического стилей мышления.

Метапредметные:

— формирование умения ориентировки в системе знаний;
— формирование умения выбора наиболее эффективных способов решения задач на компьютере в зависимости от конкретных условий;
— формирование приёмов проектной деятельности, включая умения видеть проблему, формулировать тему и цель проекта, составлять план своей деятельности, осуществлять действия по реализации плана, результат своей деятельности соотносить с целью, классифицировать, наблюдать, проводить эксперименты, делать выводы и заключения, доказывать, защищать свои идеи, оценивать результаты своей работы;
— формирование умения распределения времени;
— формирование умений успешной самопрезентации.

Формы контроля

Во время проведения курса предполагается текущий, промежуточный и итоговый контроль.

Текущий контроль осуществляется регулярно во время проведения каждого лабораторного занятия, заключается в ответе учащихся на контрольные вопросы, демонстрации полученных скриптов в среде Scratch, фронтальных опросов учителем.

Также в тематическом планировании предполагаются две промежуточные контрольные работы.

Контрольная работа для проверки полученных навыков по темам «Условный оператор if», «Циклы в языке Python»

1. Найти корни квадратного уравнения, заданного своими коэффициентами.
2. Вычислить значение функции $y = x^3 + 5x$ для заданного с клавиатуры значения аргумента x.

3. Определить, сколько отрицательных среди трёх введённых с клавиатуры чисел.
4. Проверить, принадлежит ли точка с координатами (*x* и *y*) части фигуры, изображённой на рисунке 33.

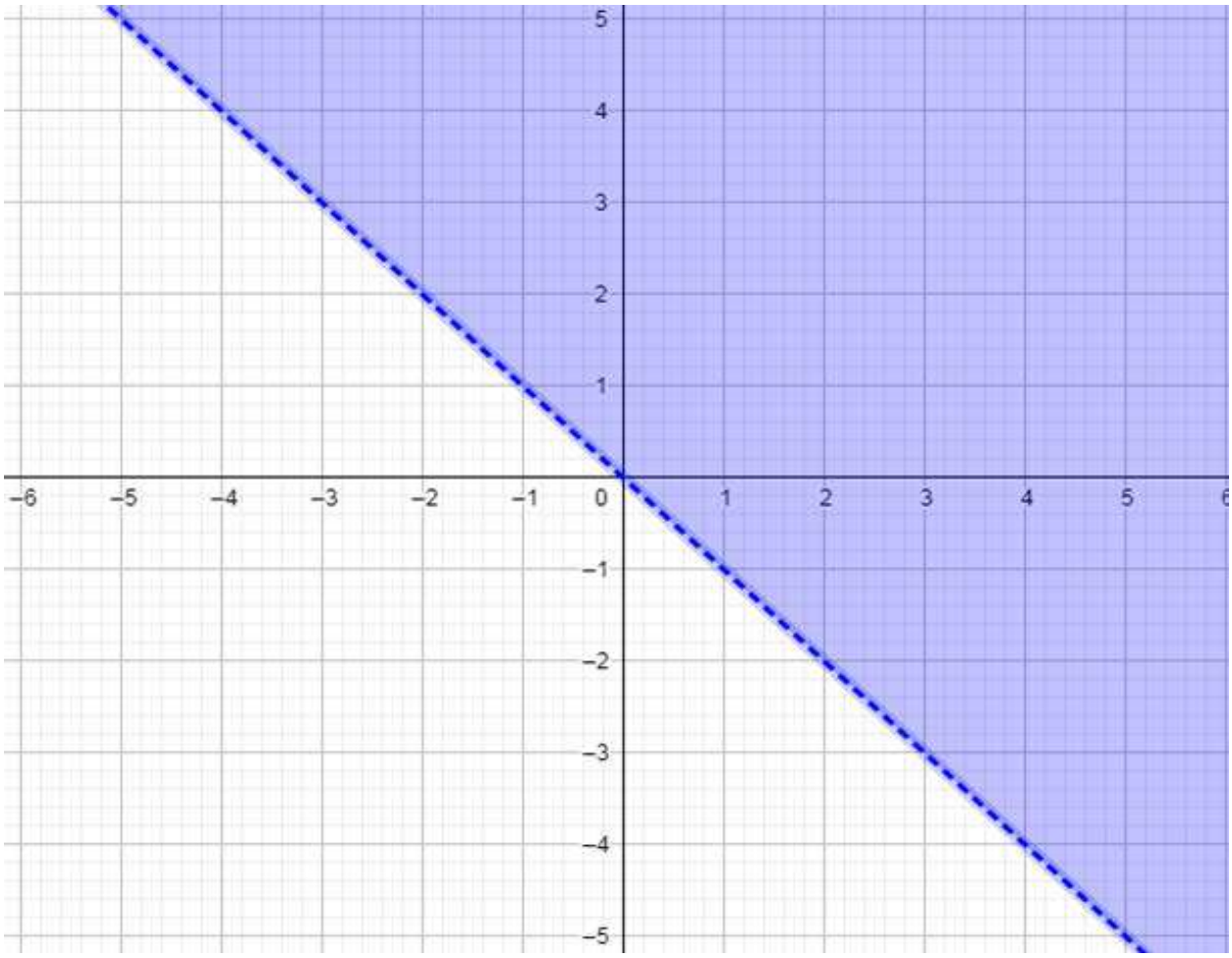

Рис. 33. Иллюстрация к задаче

5. Проверить, есть ли во введённом числе одинаковые цифры, например 121.
6. Напечатать столбиком все целые числа от 20 до 35 и вычислить корни этих чисел.
7. Найти все трёхзначные числа, которые при увеличении на 3 делятся на 5.
8. Даны натуральные числа от 20 до 50. Напечатать те из них, которые делятся на 3, но не делятся на 5.

Контрольная работа для проверки полученных навыков по темам «Списки в языке Python»

1. Поменять местами самый большой и самый маленький элементы списка.
2. Даны два списка. Получить третий список, включая в него только те элементы, которые встречаются в исходных списках только 1 раз.
3. Сформировать возрастающий список из чётных чисел данного списка.
4. Даны два списка. Удалить все элементы первого списка из второго, остальные элементы второго списка отсортировать.
5. Создать список из случайных чисел. Найти номер его последнего локального максимума (локальный максимум — это элемент, который больше любого из своих соседей).
6. Создать список из случайных чисел. Найдти второй по величине максимум.

Содержание и форма организации учебных занятий

Планы учебных занятий

1. Первые программы на языке Python, основные операторы

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение навыков по созданию первых программ в среде программирования Python, изучение основных операторов Python, ввода/вывода данных, математических операторов.

Метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные).

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ:

Изучение теоретического материала лабораторной работы, выполнение лабораторной работы 1.

2. Условный оператор if

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение навыков по использованию условного оператора if в среде программирования Python, разработка программ, реализующих разветвляющийся алгоритм.

Метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные).

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ:

Изучение теоретического материала лабораторной работы, выполнение лабораторной работы 2.

3. Циклы в языке Python

Рекомендуемое количество часов на данную тему — 4 часа.

Планируемые результаты

Предметные: получение навыков по использованию операторов цикла в среде программирования Python, разработка программ, реализующих циклический алгоритм.

Метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные).

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ:

Изучение теоретического материала лабораторной работы, выполнение лабораторной работы 3.

4. Списки в языке Python

Рекомендуемое количество часов на данную тему — 4 часа.

Планируемые результаты

Предметные: получение навыков по использованию списков в среде программирования Python, разработка программ, реализующих работу со структурами данных.

Метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные).

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ:

Изучение теоретического материала лабораторной работы, выполнение лабораторной работы 4.

5. Работа со строками в Python

Рекомендуемое количество часов на данную тему — 4 часа.

Планируемые результаты

Предметные: получение навыков по использованию строк в среде программирования Python, разработка программ, реализующих работу со строковыми данными.

Метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные).

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ:

Изучение теоретического материала лабораторной работы, выполнение лабораторной работы 5.

Описание лабораторных работ

Лабораторная работа 1. Первые программы на языке Python, основные операторы

Теоретическая часть

Предлагается рассмотреть более подробно структуру программы на языке Python, а также основные правила написания программ.

Любую программу языке Python можно представить, как набор **лексем** (допустимых символов), записанных в определённом порядке и по определённым правилам. Лексема может представлять собой: комментарии; литералы, знаки пунктуации, переменные, специальные ключевые слова.

Программа на языке Python может содержать достаточное количество комментариев, каждый комментарий начинается с символа решётка «#».

Литералы представляют собой значения, заданные в коде программы, например, числа (25) или строки (“привет”). В языке Python представлены некоторые встроенные типы объектов. В языке Python используется динамическая типизация (типы данных определяется автоматически, и их не требуется объявлять в программном коде), но при

этом он является языком со строгой типизацией (вы сможете выполнять над объектом только те операции, которые применимы к его типу).

Если говорить об использовании знаков пунктуации, то стоит отметить, что каждая строка в программе на языке Python не должна заканчиваться точкой с запятой, как, например, в C++, но если есть необходимость записать несколько инструкций в одну строчку, то их можно разделять точкой с запятой.

Переменная или идентификатор используются для хранения данных различного типа. В языке Python нет специального раздела описания переменных, в котором указывается тип переменной перед её первым использованием. Есть определённые правила для задания имён переменных языке Python: это последовательность букв, которая не может начинаться с цифры, но может содержать символ подчеркивания (_). Имена переменных чувствительны к регистру. Имена переменных не могут совпадать со специальными ключевыми словами.

Ключевые слова в языке Python имеют специальное назначение и представляют собой управляющие конструкции языка. Например: and, break, for и т. д.

При составлении программ лексемы объединяются в синтаксические конструкции, которые могут вкладываться друг в друга. В результате могут образовываться блочные конструкции, каждый блок кода начинается двоеточием (:), а тело блока выделяется обязательным отступом в виде четырёх пробелов. Обычно среда программирования сразу делает отступ для блока после двоеточия. В зависимости от используемой среды программирования блоки могут иметь визуальное выделение.

Также существует ряд рекомендаций к составлению программ на языке Python, с которыми можно ознакомится, например по ссылке https://pythonworld.ru/osnovy/sintaksis-yazyka-python.html.

Перейдём к рассмотрению основных типов оператора языка.

Операторы языка Python можно разделить на 7 типов:

— арифметические операторы;
— операторы сравнения;
— операторы присваивания;
— логические операторы;
— операторы принадлежности;
— операторы тождественности;
— битовые операторы.

Приведём примеры некоторых наиболее часто используемых операторов, приведённых в таблице ниже.

Таблица 1. Основные операторы языка Python

Оператор	Описание
+	Сложение
-	Вычитание
*	Умножение
/	Деление
//	Целочисленное деление
%	Остаток от деления
**	Возведение в степень
<	Меньше

Продолжение

Оператор	Описание
>	Больше
>=	Больше или равно
<=	Меньше или равно
==	Равно
!=	Неравно
and (И).	Логическое умножение, или конъюнкция
or (ИЛИ)	Логическое сложение, или дизъюнкция
not (НЕ)	Логическое отрицание
|	Побитовое или
^	Побитовое исключающее или
&	Побитовое и
>>	Битовый сдвиг вправо
<<	Битовый сдвиг влево
~	Инверсия битов

Рассмотрим несколько примеров использования данных операторов.

Пример 1

```
>>> 6+3
9
>>> 7**5
16807
>>> 45%4
1
```

Один из самых часто используемых операторов — оператор присваивания. Оператор присваивания в языке Python имеет вид:

<идентификатор>=<выражение>

Пример 2

```
c=6
L=121
```

Также в языке Python имеют место сложные операторы присваивания.

Таблица 2. Сложные операторы присваивания

Обозначение	Описание
+=	Левая и правая части суммируются, результат присваивается переменной в левой части
-=	Вычитается значение правой части из значения переменной в левой части, результат присваивается переменной в левой части
*=	Левая и правая части умножаются, результат присваивается переменной в левой части

Продолжение

Обозначение	Описание
/=	Делит значение переменной из левой части на значение выражения в правой части, результат присваивается переменной в левой части
%=	Находится остаток от деления значения переменной из левой части на значения выражения в правой части, результат присваивается переменной в левой части
**=	Выполняет возведение левой части в степень значения выражения из правой части, результат присваивается переменной в левой части
//=	Делит нацело значение переменной из левой части на значение выражения в правой части, результат присваивается переменной в левой части

Пример 3

```
>>> L=8
>>> L
8
>>> L+=8
>>> L
16
>>> L*6
96
>>>
```

Часто при решении задач необходимо выполнять ввод данных с клавиатуры и выводить данные на экран. Рассмотрим работу соответствующих операторов в языке Python.

Ввод данных с клавиатуры осуществляется с помощью оператора print. Формат оператора:

print()

После выполнения оператора происходит переход курсора на новую строку. Для вывода «строчку» необходимо добавить именованный end=›<разделитель> ‘.

Пример 4

```
A=9
B=14
print(A,”=”,B)
```

Результаты работы программы представлены ниже на рисунке 34:

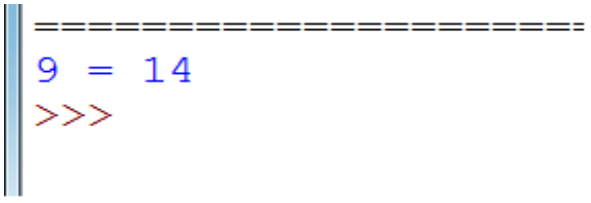

Рис. 34. Результат работы программы

Пример 5

```
Z=9
Y=15
W=-15
print(W)
print(W+Z,Y)
```

Результаты работы программы представлены ниже на рисунке 35.

Рис. 35. Результат работы программы

Пример 6

```
Z=9
Y=15
W=-15
print("W=",W, end=" ")
print(Z*10)
```

Результаты работы программы представлены ниже на рисунке 36.

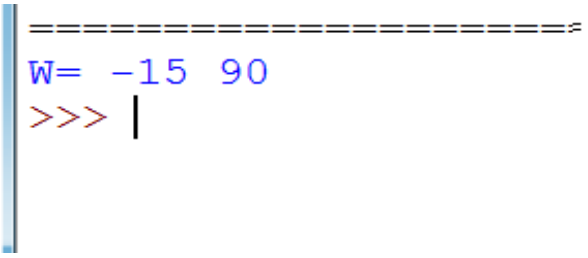

Рис. 36. Результат работы программы

Также в языке Python можно использовать форматированный вывод. Для этого необходимо использовать встроенную функцию format(). Синтаксис функции выглядит следующим образом:

<строка>.format(<формат>)

«строка» представляет собой значение для форматированного вывода, «формат» — спецификации формата ‹Mini-Language›.

Пример 7

```
N = 116
M = 130
Prod=N*M
print( “{}*{}={}”.format(N,M, Prod) )
```

Результаты работы программы представлены ниже на рисунке 37.

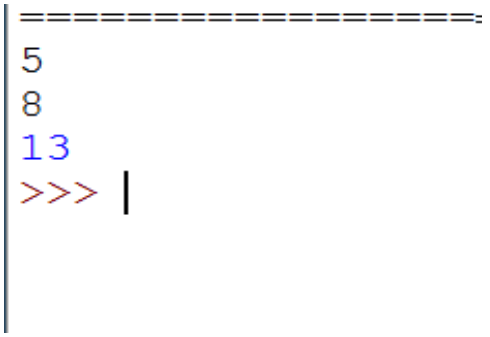

Рис. 37. Результат работы программы

Для ввода данных с клавиатуры используется встроенная функция input(). Данная функция возвращает в качестве результата строку!

В результате выполнения программы:

```
a=input()
c=input()
z=a+c
print(z)
```

на экран выводится не сумма чисел.

Результаты работы программы представлены ниже на рисунке 38.

Рис. 38. Результат работы программы

При необходимости ввода числовых значений используются встроенные функции int и float.

Функция int используется для преобразования строки в целое число. Изменим предыдущий пример:

```
n=int(input())
m=int(input())
k=n+m
print(k)
```

Результат работы — сумма чисел.

Результаты работы программы представлены ниже на рисунке 39

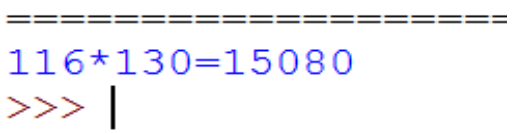

Рис. 39. Результат работы программы

Также можно указывать аргумент функции input, в этом случае на экран будет выводится «пригласительное сообщение».

```
n=int(input("n= "))
m=int(input("m= "))
k=n+m
print(k)
```

Результаты работы программы представлены ниже на рисунке 40.

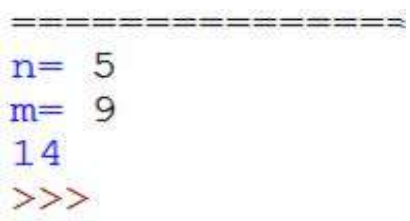

Рис. 40. Результат работы программы

Аналогично работает и функция float, но в результате её работы введённое значение преобразовывается в вещественное число.

```
n=float(input())
m=float(input())
k=n+m
print(k)
```

В этом же разделе упомянем работу со встроенными математическими функциями. Все математические функции в языке Python объединены модулем math. Данный модуль при необходимости нужно импортировать с помощью команды import:

```
import math
```

После подключения можно использовать всё множество математических функций языка Python.

В составе модуля math присутствуют несколько известных математических констант.

Таблица 3. Математические константы языка Python

Обозначение	Описание
math.pi	Математическая константа $\pi = 3.141592....$
math.e	Число Эйлера $e = 2{,}71828...$
math.inf	Положительная бесконечность

Также опишем несколько наиболее часто используемых функций, входящих в состав модуля math.

Таблица 4. Математические функции языка Python

Обозначение	Описание
math.exp(*x*)	Экспонента числа e^x
math.log(*X*, [*A*])	Логарифм *X* по основанию *A*. Если *A* не указан, вычисляется натуральный логарифм
math.pow(*X*, *n*)	Возведение в степень X^n
math.sqrt(*x*)	Квадратный корень из *x*
math.cos(*X*)	Косинус *X* (*X* указывается в радианах)
math.sin(*X*)	Синус *X* (*X* указывается в радианах)
math.tan(*X*)	Тангенс *X* (*X* указывается в радианах)
math.acos(*X*)	Арккосинус *X*. В радианах
math.asin(*X*)	Арксинус *X*. В радианах
math.atan(*X*)	Арктангенс *X*. В радианах
math.degrees(*X*)	Конвертирует радианы в градусы
math.radians(*X*)	Конвертирует градусы в радианы

Полный список математических функций можно вывести командой:
import math
>>> math
<module ‘math’ (built-in)>
>>> dir(math)

Результаты работы программы представлены ниже на рисунке 41.

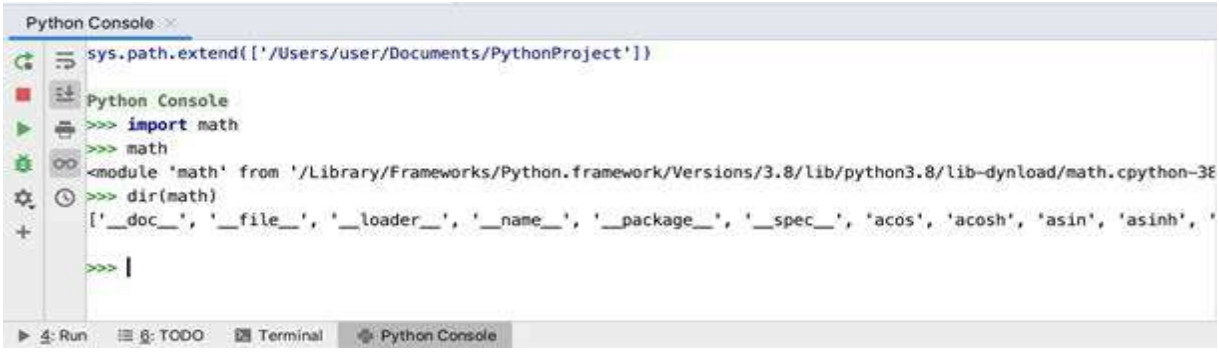

Рис. 41. Полный список математических функций языка Python

Также в языке Python есть возможность получения так называемых псевдослучайных чисел. Данные числа вычисляются по некоторой математической формуле, могут использоваться несколько алгоритмов (в языке Python используется достаточно новый алгоритм «вихрь Мерсенна»).

В языке Python для получения таких чисел необходимо обратиться к модулю random. Имеется возможность получать как целые, так и вещественные случайные числа:

— randint(*a*, *b*) — возвращает случайное целое число из отрезка [*a*; *b*];
— uniform(*a*, *b*) — возвращает случайное вещественное число из отрезка[*a*; *b*].

```
from random import randint
from random import uniform
k=randint(1, 15)
z=uniform(50, 70)
print(k)
print(z)
```

Результаты работы программы представлены ниже на рисунке 42.

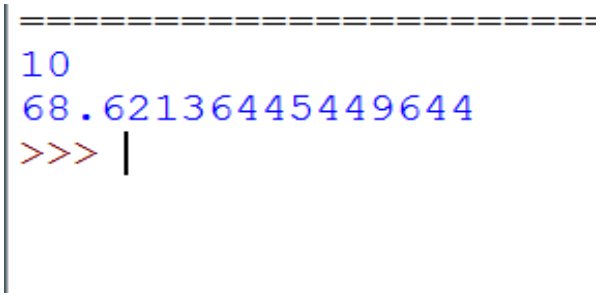

Рис. 42. Результат работы программы

Пример 8. Вычислить выражение , где *a*, *b* — целые числа, введённые с клавиатуры.

```
a=int(input(“a= “))
b=int(input(“b= “))
c=a**2+2*b
print(“c=”, c)
```

Результаты работы программы представлены ниже на рисунке 43.

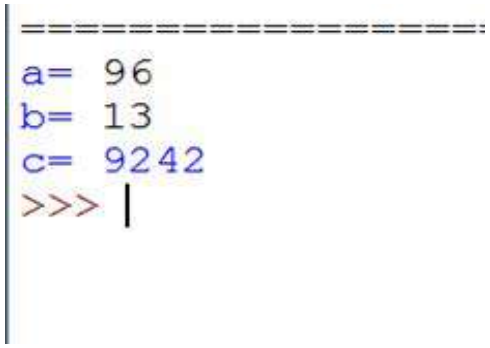

Рис. 43. Результат работы программы

Практическая часть

Цель работы: Ознакомление с основами написания программ на языке программирования Python, работа с операторами присвоения, ввода/вывода данных.

Ход лабораторной работы

1. Откройте среду разработки Python.
2. Даны длины рёбер *a, b, c* прямоугольного параллелепипеда. Найти его объём и площадь поверхности *S* (значения длин сторон вводятся с клавиатуры).

Указание. Примерный листинг программы может иметь вид:

```
a=float(input(“a= “))
b=float(input(“b= “))
c=float(input(“c= “))
print(“объем= “, a*b*c)
print(«площадь поверхности= «,2*(a*b+b*c+a*c))
```

3. Даны числа *a, b, c*. Найти их среднее арифметическое.
4. Ученик купил в магазине *n* порций мороженного по цене 300 р и *k* плиток шоколада по цене 520 руб. Сколько всего потратил учащийся на покупки?

Выводы: в ходе выполнения лабораторной работы вы получили представление о написании простых программ на языке программирования Python.

Контрольные вопросы:

1. Какие операторы языка Python вы использовали при выполнении лабораторной работы?
2. Какой оператор используется для ввода целых чисел с клавиатуры, вещественных чисел с клавиатуры?
3. Как вывести данные на экран в языке Python?

Лабораторная работа 2. Условный оператор if

Теоретическая часть

При решении задач важно реализовывать возможность выбора среди альтернативных операций на основе результатов проверки. В императивных языках программирования для этих целей используется оператор ветвления (условный оператор). В языке Python подобный оператор предусматривает не только возможность сделать выбор одной из двух альтернатив, но и предусматривает выполнение в зависимости от значения той или иной переменной одной из трёх (или более) ветвей программы.

Общая форма оператора ветвления выглядит следующим образом:

```
if <условие1>:
   оператор1
   elif <условие2>:
      оператор2
   else:
   оператор3
```

Части else и elif являются необязательными. После части if указывается логическое условие, которое может быть истинным или ложным. Опишем случаи, когда выражение является истинным:

- любое число, не равное 0, или непустой объект, а числа, равные 0, пустые объекты и значение None определяют ложь;
- операции сравнения применяются к структурам данных рекурсивно;
- операции сравнения возвращают True или False;
- логические операторы and и or возвращают истинный или ложный объект-операнд.

Логическими операндами могут быть and (конъюнкция), or (дизъюнкция), not — отрицание. Равенство обозначается == (чтобы отличить знак равенства от оператора присвоения), а неравество !=. Также в Python можно записывать двойное условие, например 2<=a<=5, -10<v<=9.

Как видно из описания оператора ветвления, он может содержать в своём составе другие операторы, в том числе другие условные. Стоит обратить внимание, что после логического условия стоит двоеточие, для того чтобы показать, что далее идёт блок выражений. Блок выражений записывается после отступа.

Рассмотрим работу оператора более подробно.

В самом простом случае оператор ветвления имеет вид:

```
if <условие1>:
   оператор1
```

Блок-схема работы данного оператора представлена на рисунке 44.

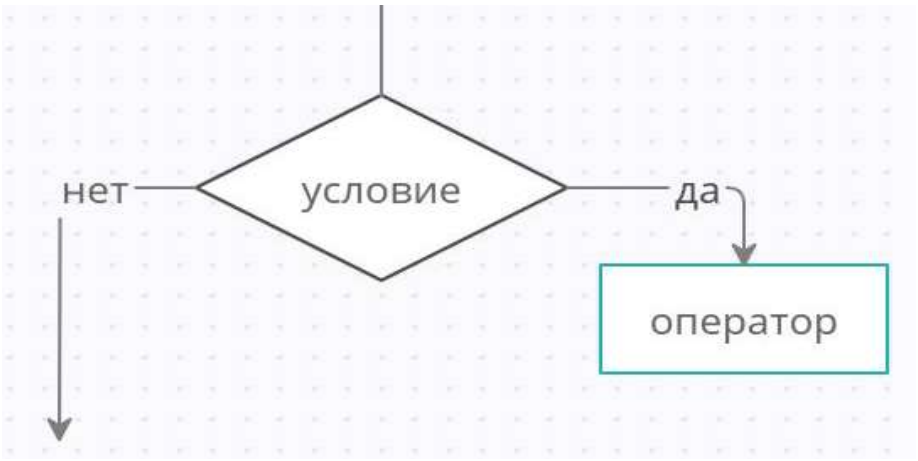

Рис. 44. Блок-схема условного оператора

В этом случае в случае истинности условия выполняется оператор, а затем осуществляется выход из оператора (управление передаётся оператору, следующему за оператором if).

Приведём пример.

Пример 1

```
a = int(input())
if a < 0:
   print('Ниже')
```

Результаты работы программы представлены ниже на рисунке 45.

Рис. 45. Результат работы программы

В данном примере в качестве условия используется сравнение a<0. Если оно выполняется, то на экран выводится «Ниже». Если же условие ложно, то программа ничего не выполняет.

Пример 2.

```
a = int(input())
if (a < 0) and(a>=-3):
   print('Ниже')
```

Результаты работы программы представлены ниже на рисунке 46.

Рис. 46. Результат работы программы

В данном примере условие составное: состоит из двух условий, объединённых операцией and (логическое и).

Рассмотрим далее более сложный вид оператора ветвления:

```
if <условие1>:
   опертор1
else:
   оператор2
```

Обратите внимание на порядок отступов в формате оператора!

Блок-схема работы данного оператора представлена на рисунке 47.

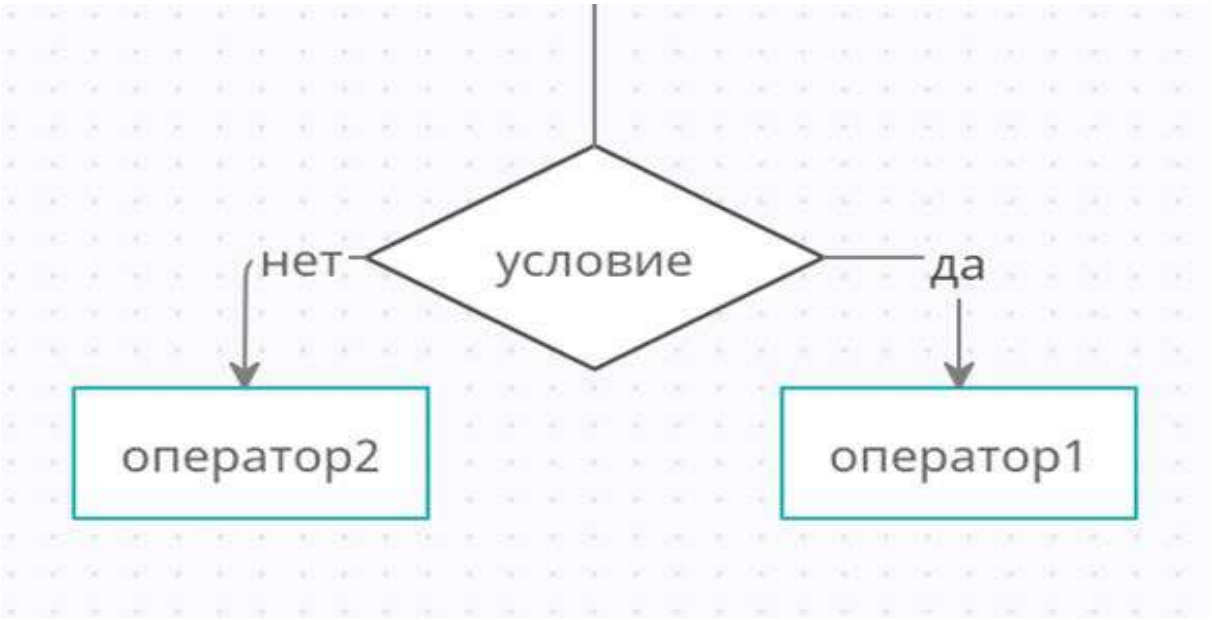

Рис. 47. Блок-схема условного оператора

Если условие истинно, то далее выполняется оператор1, а в противном случае (если условие ложно), то выполняется оператор2. А далее управление переходит к оператору, которое следует за оператором ветвления.

Приведём примеры работы более сложной формы оператор ветвления.

Пример 3

```
a = int(input())
b = int(input())
if a>2*b:
   print('да')
else:
   print('нет')
```

Результаты работы программы представлены ниже на рисунке 48.

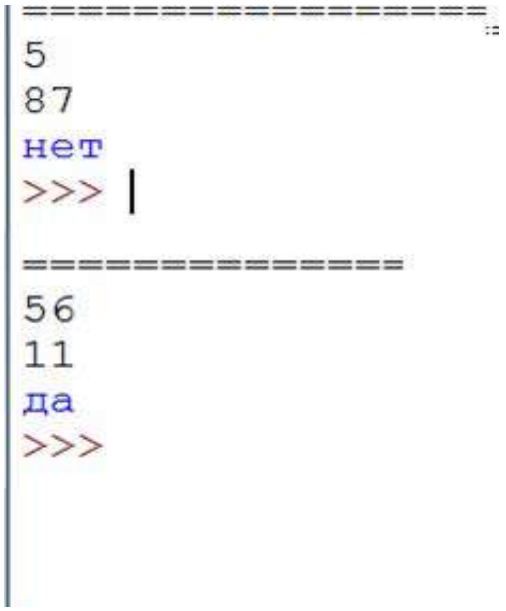

Рис. 48. Результат работы программы

В данном примере в случае истинности условия a>b выполняется оператор print(‹да'), а в противном случае — print('нет').

Обратите внимание, что после условия и после части else можно указывать несколько операторов, но тогда все они идут после отступа! Для сравнения рассмотрим два примера.

Пример 4.1

```
a = int(input())
b = int(input())
if a>2*b:
   print('да')
else:
   print('нет')
   s=a*b
print(s)
```

Результат работы данной программы представлен на рисунке 49.

```
==================
15
38
нет
570
>>>
```

Рис. 49. Результат работы программы из примера 4.1

Пример 4.2

```
a = int(input())
b = int(input())
if a>2*b:
   print('да')
else:
   print('нет')
   s=a*b
   print(s)
```

Результат работы данной программы представлен на рисунке 50.

```
------------------
13
38
нет
494
>>>
```

Рис. 50. Результат работы программы из примера 4.2

Как видим, результат работы программы отличаются.
Третья форма оператора ветвления выглядит следующим образом:

```
if <условие1>:
   оператор1
elif <условие2>:
   оператор2
else:
   оператор3
```

Блок-схема работы данного оператора представлена на рисунке 51.

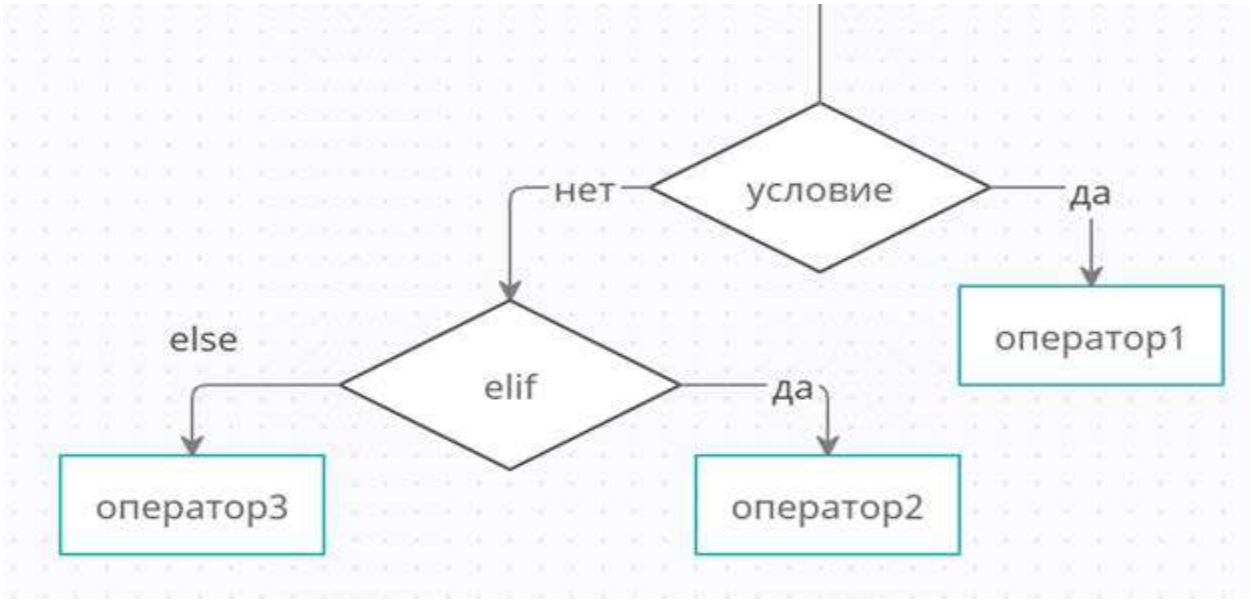

Рис. 51. Блок-схема условного оператора

При использовании данной формы можно проводить проверку нескольких условий — после if и после elif. Оператор после else выполняется в том случае, если не выполнилось условие 2 после части elif.

Рассмотрим пример использования данной формы оператора if.

Пример 5

```
a=int(input())
if a < 0:
   print(a+3)
elif -5 <= a <= 5:
   print(a*10)
else:
   print('Error!')
```

Пример 6. Написать программу вычисления стоимости покупки с учётом скидки. Скидка в 5% предоставляется в том случае, если стоимость покупки превысила 1000 р, а в 10% — если стоимость выше 3000 р.

Листинг программы представлен ниже:

```
s=int(input())
if s<1000:
   print(s)
elif 1000<=s<3000:
   print(s*0.97)
else:
   print(s*0.95)
```

Результаты работы программы представлены ниже на рисунке 52.

Рис. 52. Результат работы программы из примера 6

Ещё раз обратимся к синтаксическим правилам языка Python.

В языке Python отсутствуют фигурные скобки, как в языке C/C++, или разделители «begin/end», как в языке Pascal, окружающие блоки программного кода. Вместо этого принадлежность операторов к вложенному блоку определяется по величине отступов. Также операторы в языке Python обычно не завершаются точкой с запятой; обычно признаком конца инструкции служит конец строки с этой инструкцией.

Все составные операторы в языке Python оформляются одинаково: строка с заголовком завершается двоеточием, далее следуют одна или более вложенных инструкций, обычно с отступом относительно заголовка. Эти инструкции с отступами называются блоком (или иногда набором). Интерпретатор автоматически определяет границы блоков по величине отступов, т. е. по ширине пустого пространства слева от программного кода. Все инструкции, смещённые вправо на одинаковое расстояние, принадлежат к одному и тому же блоку кода.

В инструкции if предложения elif и else являются не только частями инструкции if, но и заголовками с собственными вложенными блоками. На рисунке 53 представлена схема с отступами для оператора if.

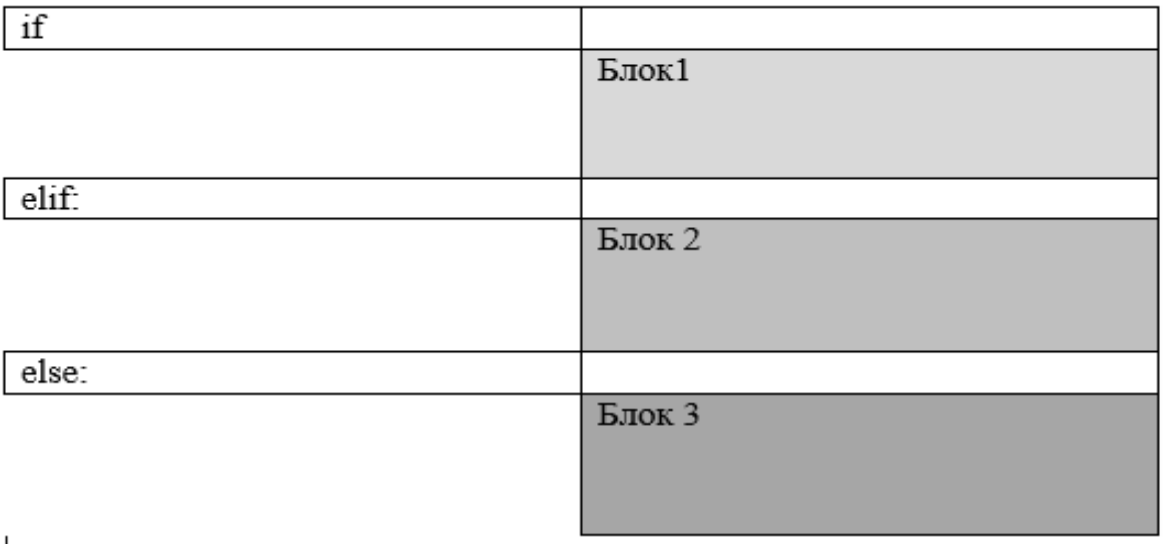

Рис. 53. Схема вложенных блоков

Практическая часть

Цель работы: ознакомление с условным оператором if на языке программирования Python.

Ход лабораторной работы

1. Откройте среду разработки Python PyCharm.
2. Составьте программу, которая запрашивает ввод температуры тела человека и определяет, здоров он или болен (нормальной температурой тела считается температура от 36,3 до 36,9).

Указание. Примерный листинг программы может иметь вид:

```
t=float(input(“t= “))
if 36.3<=t<=36.9:
   print(«Вы здоровы»)
else:
   print(«Вы больны»)
```

3. Составьте программу подсчёта площади равнобедренного треугольника. Если площадь треугольника чётная, разделить её на 2, в противном случае вывести сообщение «Не могу делить на 2!».
4. Имеется коробка со сторонами: AxBxC. Определите, пройдёт ли она в дверь с размерами MxK.
5. Определите, является ли введённый пользователем год високосным.

Указание. Примерный листинг программы может иметь вид:

```
y = int(input(«Введите год «))
if (y % 4 == 0 and y%100 != 0) or (y%400 == 0):
   print(«Высокосный»)
else:
   print(«Обычный»)
```

6. Составьте программу, проверяющую, что введённое число является чётным или нечётным.

Выводы: в ходе выполнения лабораторной работы вы получили представление о составлении условных алгоритмов с использованием оператора if-elif-else в языке программирования Python.

Контрольные вопросы:

1. Для чего используются операторы ветвления в программировании?
2. Как выглядит синтаксис оператора ветвления в языке Python?
3. Какие формы оператора ветвления можно использовать в языке Python?

Лабораторная работа 3. Циклы в языке Python

Теоретическая часть

Цикл в языке программирования представляет собой конструкцию, многократно выполняющую одну и ту же группу операторов. Число повторений циклов, или итераций, может быть задано заранее или зависеть от истинности того или иного условия. В реальной жизни постоянно применяются циклы, поэтому циклический алгоритм часто используются при решении задач по программированию.

В языке программирования Python может быть реализовано два вида цикла:
— с предусловием — цикл while;
— с параметром — цикл for.

Цикл while является одним из самых часто используемых и самых универсальных циклов в Python. Полный формат данного цикла представлен ниже:

```
while <условие>:
<оператор1>
else:
<оператор2>
```

Часть else является необязательной, она выполняется, когда управление передаётся за пределы цикла без использования инструкции break. Блок-схема работы цикла while представлена ниже на рисунке 54.

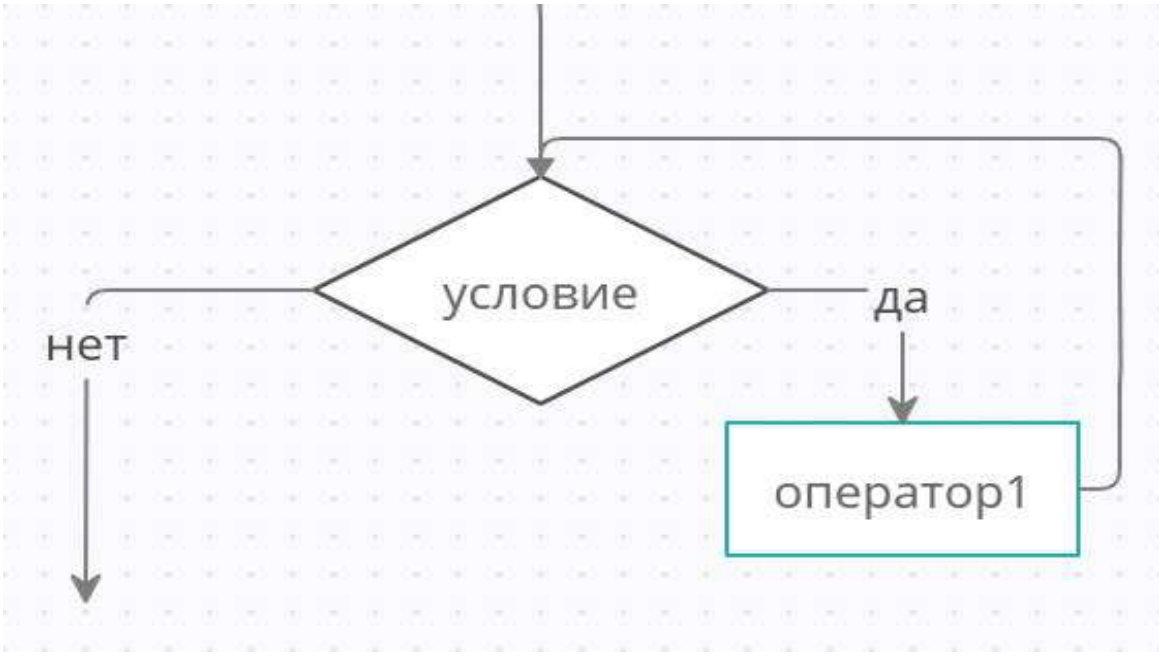

Рис. 54. Блок-схема цикла с предусловием while

Выполнение цикла while начинается с вычисления выражения. Если оно истинно (не равно false), выполняется оператор цикла. Если при первой проверке выражение равно false, цикл не выполнится ни разу. Тип выражения должен быть арифметическим или приводимым к нему. Если условие в цикле while никогда не будет ложным, то не будет причин остановки цикла и программа «зациклится». Чтобы этого не произошло, необходимо организовать момент выхода из цикла, т. е. ложность выражения в заголовке. Так, например, изменяя значение какой-нибудь переменной в теле цикла, можно довести логическое выражение до ложности. Обратите внимание, что операторы тела цикла должны быть отделены отступом.

Пример 1

```
import math
i = 5
while i < 15:
    print(math.sqrt(i))
    i+= 2
```

Интерфейс программы PyCharm с введёнными данными представлен ниже на рисунке 55.

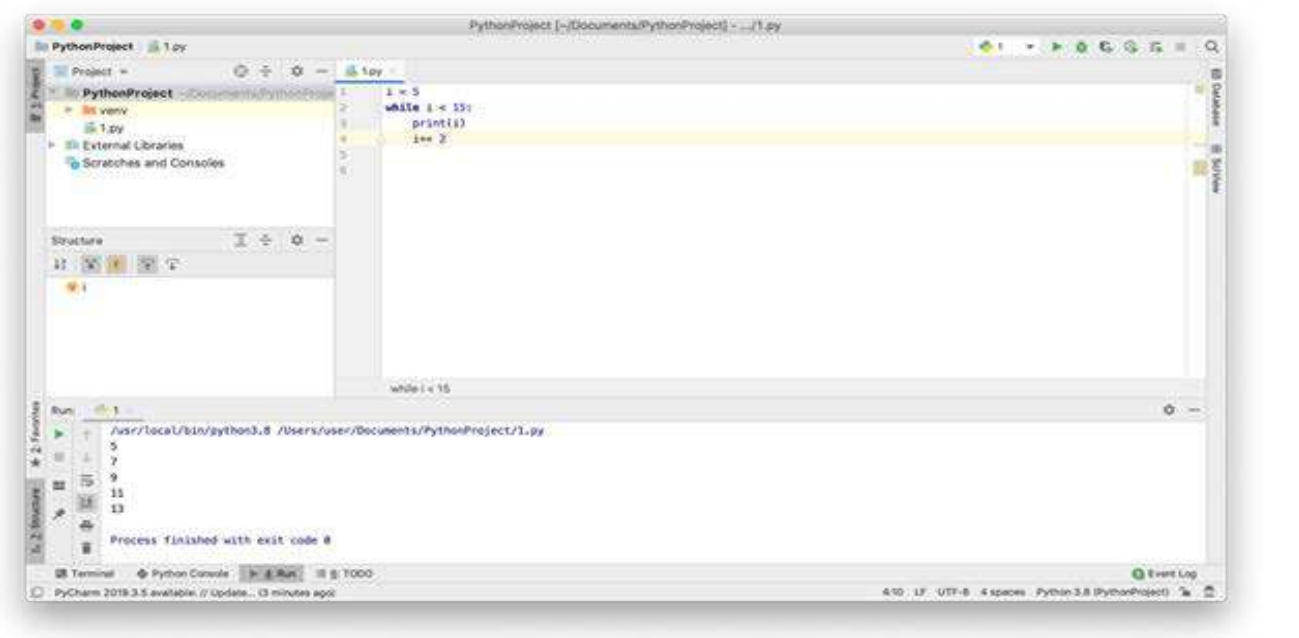

Рис. 55. Результат работы программы

В данном примере организован перебор значений переменной i с шагом 2.
Результаты работы программы представлены ниже на рисунке 56.

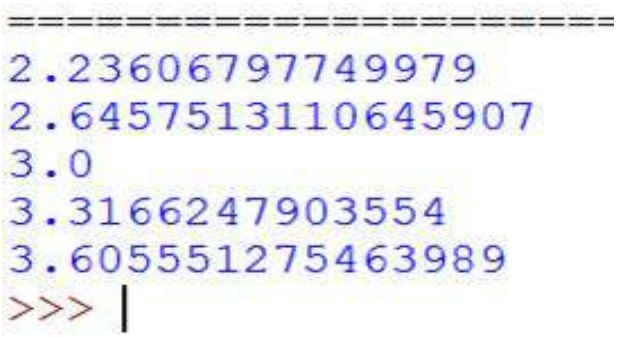

Рис. 56. Результат работы программы

2. Условие работы цикла i < 15. В теле цикла происходит изменение переменной i, поэтому цикл не будет бесконечным.

Пример 2

```
a = 0
while a < 7:
  print("Python")
  a += 1
```

В данном примере цикл выполняется, пока истинно условие $a < 7$, значение переменной a меняется в теле цикла.

Результаты работы программы представлены ниже на рисунке 57.

Рис. 57. Результат работы программы

Второй цикл, используемый в языке Python, — цикл с параметром. Синтаксис данного цикла представлен ниже:

```
for <переменная> in <объект>:
   <оператор1>
else:
   <оператор2>
```

Блок-схема работы цикла выглядит следующим образом:

Рис. 58. Блок-схема цикла с параметром

Этот цикл проходится по любому итерируемому объекту (например, строке или списку) и во время каждого прохода выполняет тело цикла заданное число раз. Для обращения к текущему элементу последовательности в теле цикла обычно используется переменная цикла, её иногда называют управляющей переменной. В цикле for также используют необязательную часть else, которая работает точно так же, как и в циклах while, — она выполняется, если выход из цикла производится не инструкцией break (о которой будет рассказано ниже).

Часто для организации работы цикла с параметром for используется функция range.

Функция range() возвращает последовательность чисел, регулируемую количеством переданных в неё аргументов. Возможны следующие варианты обращения к данной функции:

- range(finish);
- range(start, finish);
- range(start, finish, step).

Здесь start — это первый элемент последовательности (включительно), finish — последний (не включительно), а step — разность между следующим и предыдущим членами последовательности.

Например, range(5) возвращается последовательность 0, 1, 2, 3, 4. Вызов range(2 ,8) возвращается последовательность 2, 3, 4, 5, 6,7.

Рассмотрим примеры организации работы цикла с параметром.

Пример 3

```
for a in range(10):
    print(a)
```

Результаты работы программы представлены ниже на рисунке 59.

Рис. 59. Результат работы программы

В данном примере цикл выводит на экран последовательность числе от 0 до 9 включительно.

Пример 4

```
for c in range(0, 22, 3):
    print(c, end=» «)
```

В данном примере выводится на экран последовательность чисел от 0 до 21 с шагом 3.
Результаты работы программы представлены ниже на рисунке 60.

Рис. 60. Результат работы программы

Также, говоря про работу циклов в языке Python, необходимо упомянуть про операторы continue, break, else.

Оператор continue используется для перехода на следующую итерацию цикла, пропуская следующего после него операторы тела цикла.

Пример 5

```
for i in range(10):
    if i == 5:
        continue
print(i * 2, end=' ')
```

Результаты работы программы представлены ниже на рисунке 61.

Рис. 61. Результат работы программы

В данном примере при равенстве переменной i==5 используется оператор continue, в результате чего пропускается оператор print(i * 2, end=' '). Поэтому число 10 не выводится на экран.

Оператор break используется для организации немедленного выхода из цикла. То есть происходит досрочное завершение работы цикла.

Пример 6

```
for i in range(10):
    if i == 5:
        break
    print(i * 2, end=' ')
```

Результаты работы программы представлены ниже на рисунке 62.

Рис. 62. Результат работы программы

В данном примере при равенстве переменной i==5 используется оператор break, в результате чего происходит завершение работы цикла. То есть последнее значение, рассмотренное в теле цикла, будет i=4.

Оператор else используется для проверки, был ли произведён выход из цикла инструкцией break или же цикл завершился образом.

Пример 7

```
for i in range(10):
    if i == 20:
        continue
    print(i * 2, end=' ')
else:
    print(«значение не найдено»)
```

Результаты работы программы представлены ниже на рисунке 63.

Рис. 63. Результат работы программы

В данном примере после вывода на экран последовательности от 0 до 18 на экран также выводится строка «значение не найдено», так как оператор continue не сработал.

Также в языке Python возможно использование вложенных циклов, когда есть один внешний цикл и один или несколько вложенных. Стоит отметить, что использование вложенных циклов может замедлить работу программы.

Приведём ещё несколько примеров работы с циклами for и while.

Пример 8. Вывести на экран десять первых степеней числа 2.

```
import math
for i in range(1,11):
    print(«2^{}={}».format(i,2**i))
```

Результаты работы программы представлены на рисунке 64.

В данной задаче используется цикл for, так как известно количество повторов цикла — *k* дней. Внутри цикла идёт увеличение переменной *n*, обозначающей длину дистанции, n+=n**0.1.

```
==========
2^1=2
2^2=4
2^3=8
2^4=16
2^5=32
2^6=64
2^7=128
2^8=256
2^9=512
2^10=1024
>>>
```

Рис. 64. Результат работы программы

Пример 9. Вывести на экран все числа из отрезка [0,999], сумма цифр которых равна введённом числу *n*.

```
n=int(input("n= "))
for i in range (0,1000):
    sum=0
    k=i
    while k!=0:
        sum+=k%10
        k//=10
if sum==n:
    print(i)
```

Результаты работы программы представлены ниже на рисунке 65.

```
==========
n= 12
39
48
57
66
75
84
93
129
138
147
156
165
174
183
192
219
228
237
```

Рис. 65. Результат работы программы

Пример 10

Разложить натуральное число на простые множители.

```
k=int(input(«Введите число «))
print(k,'= ')
l=2
while not(k==1):
    if k%l==0:
        k=k/l
        print(l, end=' ')
    else:
        l+=1
```

Результаты работы программы представлены ниже на рисунке 66.

Рис. 66. Результат работы программы

В данном примере первое простое число l=2. В цикле while введённое *k* несколько раз делится на потенциальный простой делитель l, если целочисленное деление не может быть выполнено, то ищется следующий простой делитель.

Практическая часть

Цель работы: ознакомление с операторами цикла while, for на языке программирования Python.

Ход лабораторной работы

1. Откройте среду разработки Python PyCharm.
2. Население города на 2000 г. насчитывало 620 тыс. человек. Считая темп прироста населения за год равным 3,7%, определите, в каком году оно превысит 1,5 млн человек.
3. Среди введённых пользователем *n* чисел выведите те, которые кратны 5.
4. Найдите все натуральные числа из промежутка [1; 200], у которых количество делителей равно *n* (где *n* вводится с клавиатуры).

Указание. Примерный листинг программы может иметь вид:

```
n=int(input(«Введите кол-во делителей «))
for i in range(1,201):
  k=2
  for j in range (2,i):
    if i%j==0:
      k+=1
  if k==n:
    print(i)
```

5. Выведите на экран все «счастливые» шестизначные числа.
6. Найдите все трёхзначные числа, которые являются палиндромами.

Выводы: в ходе выполнения лабораторной работы вы получили представление о составлении циклических алгоритмов с использованием оператора while, for в языке программирования Python.

Контрольные вопросы:

1. Для чего используются циклы в языке программирования?
2. Какие виды циклов реализованы в языке Python?

3. Каков синтаксис оператора цикла while?
4. Каков синтаксис оператора цикла for?
5. Для чего используется оператор break внутри тела цикла?

Лабораторная работа 4. Списки в языке Python

Теоретическая часть

Список в языке Python представляет собой структуру для хранения объектов различных типов. Элементы одного списка могут иметь одинаковый тип, но допускаются и смешанные списки. Элементы списка заключаются в квадратные скобки. Все элементы проиндексированы (пронумерованы) начиная с 0.

Примеры списков выглядят следующим образом:

Таблица 5. Примеры списков

Список	Описание
L=[]	Пустой список
L=[5, 8, 21, 6, 16]	Четыре элемента с индексами 0..4
L = [‘abc’, [‘123’, ‘xyz’]]	Вложенные списки

Списки в Python, с одной стороны, похожи на массивы в других языках программирования, например Pascal, но с другой стороны, имеется ряд отличий. Например, Python размещает элементы списка в памяти, затем размещает указатели на эти элементы. Таким образом, список в Python — это массив указателей.

Опишем ряд существенных свойств списков в языке Python.

Как было сказано ранее, можно обращаться к элементу списка по его индексу, начиная с 0. В языке Python также используют отрицательную индексацию, которая начинается с конца, с индекса —1. На рисунке 67 представлены различные виды индексации элементов списка.

L	7	8	-13	3
Индекс	0	1	2	3
Отрицательный индекс	-4	-3	-2	-1

Рис. 67. Индексация элементов списка

Например, элемент L[3] равен 3, а элемент L[—4] равен 7.

Список в языке Python может иметь переменную длину.

Далее рассмотрим способы задания списка в языке Python.

Можно непосредственно задать список, перечислив его элементы в квадратных скобках через запятую, l=[4, —3,6].

Также можно использовать функцию range, описанную ранее: a = list (range (20)). В результате сформируется список значений от 0 до 19.

Для задания элементов списка случайным образом можно использовать функцию randint из модуля random.

Пример 1

```
from random import randint
a = [randint(-10, 10) for i in range(6)]
print(a)
```

В данном примере формируется список из двадцати элементов, заданных случайным образом.

Результаты работы программы представлены ниже на рисунке 68.

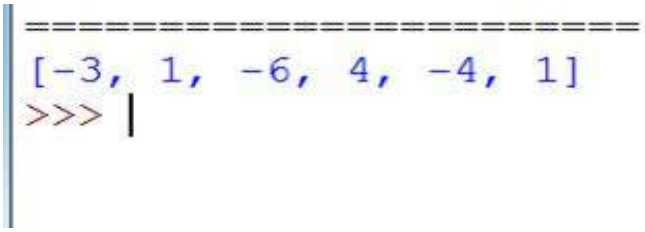

Рис. 68. Вид списка, заданного случайным образом

Ещё один вариант — задать список с клавиатуры a = [input () for i in range (10)]. Или для формирования списка с клавиатуры использовать следующий фрагмент.

Пример 2

```
a = []
n = int(input())
for i in range(n):
    k = int(input())
    a.append(k)
print(a)
```

Результаты работы программы представлены ниже на рисунке 69.

Рис. 69. Результат работы программы

Как видно из последнего примера, для обработки элементов списка часто используют циклы, в частности цикл for.

Для получения доступа к элементам списка можно использовать следующие конструкции.

Пример 3

```
for x in [“123”, “456”, “789”]:
   print(x)
```

или

```
l=[“abc”, “cde”, “efg”]
   for x in l:
      print(x)
```

Результаты работы программы представлены ниже на рисунке 70.

Рис. 70. Результат работы программы

При работе со списками в языке Python часто используют **срезу**.

Срезы используются, чтобы обрезать список, взяв лишь те элементы, которые нам будут нужны. Они работают по следующей схеме:

list[начало:конец:шаг], где:

— **Начало** указывает, с какого элемента стоит начать (по умолчанию равна 0);
— **Конец** указывает, по какой элемент берутся элементы (по умолчанию равно длине списка);
— задаёт значение, с каким шагом берём элементы, к примеру, каждый 2 или 3 (по умолчанию каждый 1).

Приведём примеры работы со срезами.

Пример 4

```
l=[2,4,6,8,10,12,14,16]
print(l[2::2])
```

Результаты работы программы представлены ниже на рисунке 71.

Рис. 71. Результат работы программы

В результате из списка берётся каждый второй элемент, начиная со второго.

Пример 5

```
l=[2,4,6,8,10,12,14,16]
print(l[::3])
```

Результаты работы программы представлены ниже на рисунке 72.

Рис. 72. Результат работы программы

В результате из списка берётся каждый третий элемент.
Также с помощью срезов можно менять и существующие списки.

Пример 6

```
l=[2,4,6,8,10,12,14,16]
l =l[1:6:1]
print(l)
```

Результаты работы программы представлены ниже на рисунке 73.

Рис. 73. Результат работы программы

В результате в списке остаются только со второго по шестой элементы.

Также в языке Python доступно достаточно много встроенных функций по обработке списков. Некоторые из них описаны в таблице 6.

Таблица 6. Описание основных функций (методов) по обработке списков

Наименование	Описание
append(x)	Добавляет элемент *x* в конец списка
clear()	Очищает список
copy()	Возвращает копию списка
count(x)	Возвращает количество элементов со значением *x*, входящих в список
extend(L)	Расширяет список через добавление в него всех элементов списка L
index(x, [start [, end]])	Возвращает индекс в списке первого элемента со значением *x* (при этом поиск ведется от start до end). Если start и end не указываются, то начиная с нулевой позиции
insert(i, x)	Вставляет в список на i-ю позицию значение *x*
pop([i])	Удаляет из списка i-й элемент и возвращает его. Если индекс не указан, удаляется последний элемент
remove(x)	Удаляет первый элемент в списке, имеющий значение *x*. Возвращает ValueError, если такого элемента не существует
reverse()	Переворачивает список
sort()	Сортирует список

Поскольку данные функции являются методами класса «Список», то обращение к ним имеет вид:

< имя списка>. <имя метода>

Рассмотрим несколько примеров по работе с описанными функциями

Пример 7

```
l=[2,4,6,8,10,12,14,16]
print(l.index(4))
```

В результате на экран выводится индекс элемента, равного 4. В данном случае это 1.

Пример 8

```
l=[2,4,6,8,10,12,14,16]
print(l.count(6))
```

В результате на экран выводится количество элементов в списке, равных 12. В данном случае — один раз.

Пример 9

```
l=[2,4,6,8,10,12,14,16]
l.append(4)
print(l)
```

Результаты работы программы представлены ниже на рисунке 90:

В результате в конец списка добавляется новый элемент — 0.

Пример 10

```
l=[2,4,6,8,10,12,14,16]
z=[3,7,10]
l.extend(z)
print(l)
```

Результаты работы программы представлены ниже на рисунке 74.

```
========== RESTART: E:/ТочкиРо
[2, 4, 6, 8, 10, 12, 14, 16, 3, 7, 10]
>>>
```

Рис. 74. Результат работы программы

В результате список l расширяется за счёт добавления элементов списка z.

Далее рассмотрим несколько примеров задач, при решении которых используются списки.

Пример 11

Задать числовой список случайным образом. Необходимо вывести для каждого элемента сумму двух соседних с ним элементов. Для элементов списка, являющихся крайними, одним из соседей считается элемент, находящий на противоположном конце этого списка.

```
from random import randint
z = [randint(-10, 10) for i in range(10)]
print(z)
print("новый список:")
n=[]
n.append(z[1] + z[len(z) — 1])
for i in range(1, len(z) — 1):
   n.append(z[i-1] + z[i+1])
new.append(z[len(z) — 2] + z[0])
print(n)
```

В данном примере задан список из 10 элементов, диапазон элементов списка — от −10 до 10. Затем элементы списка заменяются с использованием перебора элементов списка с помощью цикла for.

Пример 12. Задан числовой список. Удалить из него все вхождения числа, введённого с клавиатуры.

```
c = [3,8,33,-12,16,3,-9,16]
k=int(input())
i=0
while i <len(c):
   if c.count(k)>1:
      c.remove(k)
   i+=1
print(c)
```

Результаты работы программы представлены ниже на рисунке 75.

```
========== RESTART:
5
[3, 8, 33, -12, 16, 3, -9, 16]
>>>
```

Рис. 75. Результат работы программы

В данном примере используются несколько встроенных функций для работы со списками: count, len, remove. Count возвращает количество вхождений элемента *k* в этот же список. Если количество вхождений больше одного, то элемент удаляется из списка.

Пример 13

Даны два числовых списка. Вывести на экран элементы, которые содержатся в обоих списках.

```
L1=[1,4,9,16,25,36,49,64]
L2=[1,2,3,4,5]
for i in range(0,len(L1)):
    if L2.count(L1[i])!=0:
        print(L1[i])
```

Результаты работы программы представлены ниже на рисунке 76.

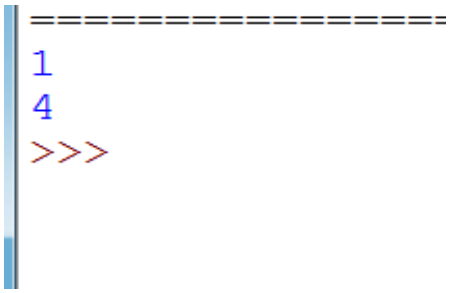

Рис. 76. Результат работы программы

В данном примере используются несколько встроенных функций для работы со списками: count, len. С помощью функции count находится количество вхождений элемента списка L1 в список L2. Если это количество не равно 0 (т. е. элемент входит в список L2), то элемент выводится на экран.

Практическая часть

Цель работы: ознакомление с понятием «список» в языке программирования Python.

Ход лабораторной работы

1. Откройте среду разработки Python.
2. Дан список некоторых целых чисел. Найдите значение 20 в нём и, если оно присутствует, замените его на 200.
3. Напишите программу, которая выводит чётные числа из заданного списка и останавливается, если встречает число 15.

Указание. Примерный листинг программы может иметь вид:

```
from random import randint
z = [randint(-10, 10) for i in range(10)]
print(z)
for x in z:
    if x==15:
        break
    if x%2==0:
print(x)
```

4. Замените в массиве нулевые элементы единицами.
5. Дан массив из 20 элементов. Сформируйте новый массив, поместив в него только те элементы, которые не превосходят введённого с клавиатуры числа.

6. Найдите в списке наибольший и наименьший элементы. Выведите их индексы на экран.

Указание. Примерный листинг программы может иметь вид:

```
from random import randint
z = [randint(-10, 10) for i in range(10)]
print(z)
m1=max(z)
print("max=",m1)
m2=min(z)
print("min=",m2)
index_max=z.index(m1)
print("индекс max=",index_max)
index_min=z.index(m2)
print("индекс min=",index_min)
```

Решение данной задачи осуществлено только за счёт встроенных функций языка Python по работе со списками. Максимальный элемент и его индекс были найдены с помощью функции max и index, аналогично найдены минимальный элемент и его индекс.

Можно ли выполнить решение данной задачи без встроенных функций?

Такое решение будет напоминать решение в другом императивном языке программирования, например Pascal. Приведём ниже пример такого решения.

```
from random import randint
z = [randint(-10, 10) for i in range(10)]
print(z)
m1=z[0]
for x in z:
    if x>m1:
        m1=x
print("max=",m1)
for i in range (0, len(z)):
    if z[i]==m1:
        index_max=i
print("индекс max=",index_max)
m2=z[0]
index_min=0
for i in range(0, len(z)):
    if z[i]<m2:
        m2=z[i]
        index_min=i
    print("min=",m2)
print("индекс min=",index_min)
```

Как видим, второй вариант решения является более длинным и выполняться, соответственно, будет дольше, за счёт нескольких обращений к циклу for.

7. Найти сумму элементов массива, находящихся между наибольшим и наименьшим.

Указание. Примерный листинг программы может иметь вид:

```
from random import randint
z = [randint(-10, 10) for i in range(10)]
print(z)
index_max=z.index(max(z))
```

```
index_min=z.index(min(z))
s=0
if index_max>index_min:
   for i in range(index_min+1,index_max):
      s+=z[i]
else:
   for i in range(index_max+1,index_min):
      s+=z[i]
print(“сумма=”,s)
```

Выводы: в ходе выполнения лабораторной работы вы получили представление о работе со списками в языке программирования Python.

Контрольные вопросы:

1. Дайте определение списка в языке программирования Python. Приведите примеры списков.
2. Как можно обратиться к элементу списка в языке программирования Python?
3. Для чего используется срез при работе со списками в языке Python?
4. Какие основные встроенные функции по работе со списками в языке Python вы можете назвать?

Лабораторная работа 5. Работа со строками в Python

Теоретическая часть

Строковые данные используются в языке Python для записи текстовой информации. Строки в Python представляют собой упорядоченные последовательности символов, используемые для хранения и представления текстовой информации.

Каждый символ имеет порядковые номера в таблице, которая называется кодовой таблицей. По умолчанию в языке Python используется стандарт «Unicode». Строки в Python обрамляются кавычками или апострофами, при этом важно, чтобы с обоих концов строки использовались кавычки одного и того же типа. Важно отметить, что в языке Python не используется тип char, привычный для многих языков программирования, например Pascal, C++. То есть один символ, заключённый в апострофы, также представляет собой строку.

Приведём примеры задания строк.

Таблица 7. Примеры строк

Список	Описание
S=’’	Пустая строка
S=’A’	Строка, состоящая из одного символа
S=’Hello’	Строка, состоящая из нескольких символов, заключённых в апострофы
S=”good”	Строка, состоящая из нескольких символов, заключённых в кавычки

Также в строках в языке Python могут использоваться специальные непечатаемые символы. Ниже в таблице представлен перечень таких символов, которые носят название

экранированные последовательности. Экранированные последовательности — это последовательности, которые начинаются с символа «\», за которым следует один или более символов.

Таблица 8. Экранированные последовательности

Экранированные последовательности	Описание
\n	Перевод строки
\t	Горизонтальная табуляция
\v	Вертикальная табуляция
\r	Возврат каретки в начало строки

Например, для фрагмента

```
s=»abc \n efg»
print(s)
```

На экран выводиться следующее (рис. 77).

Рис. 77. Результат работы программы

Аналогично спискам, строки в языке Python являются упорядоченными последовательностями символьных данных и, следовательно, могут быть проиндексированы. Для обращения к отдельному символу в строке можно указать имя строки, за которым следует число (индекс) в квадратных скобках []. Так же как и в списках, нумерация символов начинается с 0 индекса. Например: s=”hello”, s[2] возвращает символ l. Индексы также могут быть представлены и отрицательными значениями (рис. 78).

Строка S	u	n	i	t
Индекс	0	1	2	3
Отрицательный индекс	-4	-3	-2	-1

Рис. 78. Иллюстрация индексирования символов строки

Следует обратить внимание, что строки в языке Python, в отличие от списков, являются **неизменяемыми**! После того как строка будет создана, её нельзя изменить, т. е. все операции над строками в результате создают новую строку. Например, нельзя изменить отдельный символ строки: обращение s[3]='x' вызовет ошибку.

Так же как и для списков, для строк в языке Python доступны различные срезы. Формат срезы строк схож с форматом среза списков:

str[начало:конец:шаг]

— **Начало** указывает, с какого элемента стоит начать (по умолчанию равна 0);
— **Конец** указывает, по какой элемент берутся элементы (по умолчанию равно длине строки);
— **Шаг** задаёт значение, с каким шагом берём элементы, к примеру, каждый 2 или 3 (по умолчанию каждый 1).

Приведём несколько примеров обращения со срезами строк.

Пример 1

```
word = 'university'
print(word[4:6])
```

Выводит на экран два символа.

Пример 2

```
word = 'strength'
print(word[:2])
```

Выводит на экран два символа.

Результаты работы программы представлены ниже на рисунке 79.

Рис. 79. Результат работы программы

Рассмотрим основные операции над строками, разрешённые в языке Python (табл. 9).

Таблица 9. Операции над строками

Операция	Описание	Пример
+	Сложение (конкатенация) строк. В результате применения возвращается строка, равная «склейке» указанных строк	a= 'py' b = 'th' b = 'on' s=a+b+c

Продолжение

Операция	Описание	Пример
*	Умножение строк. Оператор создаёт несколько копий строки, формат операции *s*n* или *n*s*, где *s* — это строка, а *n* — натуральное число	s=’ab’ sn=s*4
in	Оператор принадлежности, который возвращает True, если подстрока входит в строку, и False, если нет	If ‘z’ in s: print(5)
>, <, >=, <=, ==, !=.	Сравнение строк	S1=”ab S2=”xy” If S1>S2: print(“Ok”)

Пример 3

```
s1=”abc”
s2=”123”
s3=s1+s2
s4=s1*3
print(s3,’ ‘,s4)
if s3>s4:
   print(s3)
else:
print(s4)
```

Результат работы программы представлен ниже (рис. 80).

Рис. 80. Результат работы программы

Также в языке Python используется достаточное число встроенных функций по обработке строк. Опишем некоторые из них в таблице 10.

Таблица 10. Описание основных функций (методов) по обработке строк

Наименование	Описание
ord(x)	Код символа *x* в ASCII

Продолжение

Наименование	Описание
chr(n)	Возвращает символ, код которого в ASCII *n*
len()	Возвращает длину строки
isnumeric()	Возвращает True, если строка представляет собой число
lower()	Переводит строку в нижний регистр
upper()	Переводит строку в верхний регистр
find(str[,start [, end]])	Возвращает индекс подстроки в строке. Если подстрока не найдена, возвращается число −1
replace(old, new[, num])	Заменяет в строке одну подстроку old на другую — new. Num ограничивает количество замен
split([d[, num]])	Разбивает строку на подстроки в зависимости от разделителя d. Num определяет максимальное количество частей для разбиения
join(strs)	Объединяет строки в одну строку, вставляя между ними определённый разделитель strs

Полный список методов строк можно увидеть, если вызвать функцию dir().

Пример 4. Дана строка. Разделите её на две равные части следующим образом: если длина строки чётная, то разделить пополам, а если длина строки нечётная, то длина первой части должна быть на один символ больше. Переставьте эти две части местами, результат запишите в новую строку и выведите на экран.

```
a=input()
s=len(a)
a1=''
if s%2==0:
    a1=a[s//2::]+a[0:s//2]
else:
    s+=1
    a1=a[s//2::]+a[0:s//2]
print(a1)
```

В данном примере используется встроенная функция replace, которая заменяет все требуемые по условию символы.

Пример 5. В строке заменить пробелы знаком тире —. Если встречается подряд несколько пробелов, то их следует заменить одним знаком «—», пробелы в начале и конце строки удалить

```
s = input(«Введите строку: «)
i = 0
while s[i] == ' ':
    i+=1
s = s[i:]
i = len(s)
while s[i-1] == ' ':
    i-=1
s = s[:i]
sn = s[0]
```

```
i = 1
while i < len(s):
   if s[i] != ' ':
      sn += s[i]
   elif s[i-1] != ' ':
      sn += '-'
   i += 1
print(sn)
```

Результаты работы программы представлены ниже на рисунке 81.

Рис. 81. Результат работы программы

В данном примере используется встроенная функция len, которая возвращает длину строки, также используются индексы для обращения к отдельному символу строки. В данном примере не рекомендуется использовать функцию replace, так как она заменит все пробелы на тире, что противоречит условию задачи.

Пример 6. В данной строке заменить все вхождения символа 1 на “one”.

```
s = input(“Введите строку: “)
sp=s.replace(‘1’,’one’)
print(sp)
```

Результаты работы программы представлены ниже на рисунке 82.

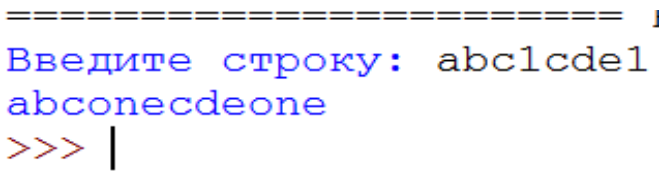

Рис. 82. Результат работы программы

Практическая часть

Цель работы: ознакомление с понятием «строка» в языке программирования Python.

Ход лабораторной работы

1. Откройте среду разработки Python.
2. Замените в строке все символы ‘a’ на ‘mama’.

3. Подсчитайте, сколько раз в строке встречаются символы +, –, *, /, % во введённой строке.

Указание. Примерный листинг программы может иметь вид:

```
s = input(«Введите строку: «)
s1=’+-*/%’
k=0
for c in s:
   if c in s1:
      k+=1
print(k)
```

4. Отредактируйте предложение, удаляя из него лишние пробелы, оставляя только по одному пробелу между словами.

5. Условимся, что использование символа ‘#’ обозначает удаление предыдущего символа. Отредактируйте строку, удалив лишние символы.

6. Подсчитайте количество гласных букв в строке.

Указание. Примерный листинг программы может иметь вид:

```
s = input(«Введите строку: «)
k=0
gl=’аоеиуыеэ’
for x in s:
   if x in gl:
      k+=1
print(“кол-во гласных “, k)
```

7. Найдите процентное содержание цифр в исходном тексте.

8. Найдите сумму чисел, встречающихся в строке.

Выводы: в ходе выполнения лабораторной работы вы получили представление о работе со строками в языке программирования Python.

Контрольные вопросы:

1. Дайте определение строки в языке программирования Python. Строки должны заключаться в кавычки или апострофы?

2. Можно ли изменять строку в языке программирования Python? Отдельный символ строки?

3. Какие основные встроенные функции по работе со строками в языке Python вы можете назвать?

Дидактические материалы

Работа со средой программирования Python

В данной работе предполагается опираться на Python версии 3.x, если в вашей операционной системе уже установлен Python версии 2.x, то эти две версии могут быть одновременно установлены в одной операционной системе и можно по выбору использовать ту или иную версию в зависимости от своих потребностей.

Далее рассматривается установка Python версии 3.x.

Предлагается рассмотреть установку Python в операционных системах семейства Windows.

Для установки нужно в браузере перейти на официальную страницу http://www.python.org/download/ и загрузить последнюю версию Python 3. Установка производится стандартным способом, так же как и для любых других программ Windows.

Для того чтобы работать с Python из командной строки Windows, необходимо установить переменную PATH. Это можно сделать при установке, выбрав соответствующий пункт Add Python 3.x to PATH (рис. 83).

Рис. 82. Окно установки Python 3.x для Windows

Для её настройки необходимо перейти в свойства компьютера или нажать кнопку «Пуск», далее выбрать «Панель управления», далее «Система и безопасность», далее «Система». Потом нажать кнопку «Дополнительные параметры системы» слева, а затем выбрать вкладку «Дополнительно». Внизу нажать кнопку «Переменные среды» и в отделе «Системные переменные» найти переменную PATH, выбрать её и нажать «Редактировать».

Далее перейти к концу строки в поле «Значение переменной» и дописать: C:\Python3x. Где x — это номер версии. В нашем случае это Python3.8 значит, дописать нужно: C:\Python38.

Если значение переменной было %SystemRoot%\system32;, теперь оно примет вид %SystemRoot%\system32;C:\Python3x. Если все шаги установки выполнены правильно, то можно запускать интерпретатор из командной строки в Windows. Чтобы открыть терминал в Windows, нажмите кнопку «Пуск» и выберите «Выполнить». В появившемся диалоговом окне наберите cmd и нажмите Enter. Далее набираем слово python и проверяем, нет ли ошибок (рис.84).

Рис. 84. Работа Python 3.x в командной строке Windows

Для пользователей Windows, чтобы запустить интерпретатор в командной строке, нужно правильно установить переменную PATH. Для открытия командной строки в Windows нужно перейти в меню «Пуск» и нажать «Выполнить...». В появившемся диалоговом окне ввести «cmd» и нажать клавишу Enter; теперь у вас будет всё необходимое для начала работы с python3 в командной строке.

Альтернативным способом можно использовать IDLE, которая устанавливается вместе с языком Python, необходимо перейти «Пуск», далее «Программы», далее «Python 3.x» и выбрать «IDLE (Python GUI)» (рис. 85).

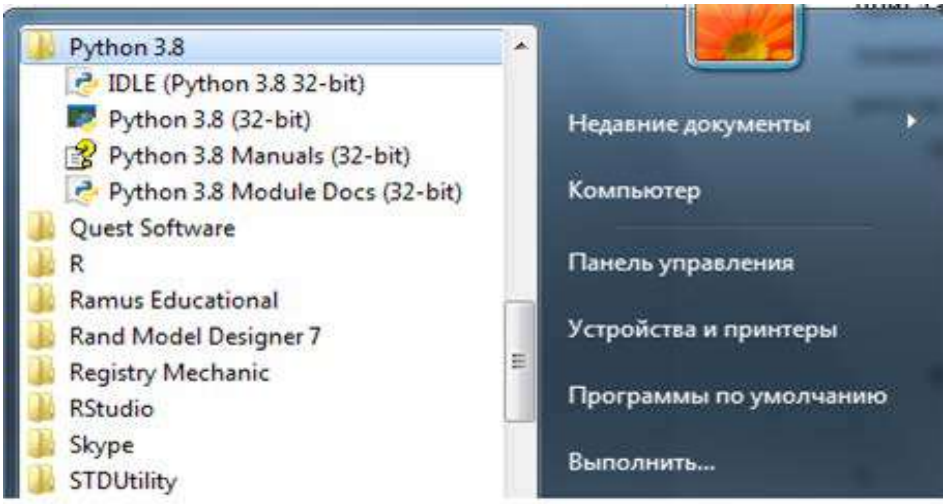

Рис. 85. Запуск Python 3.x

Как только будет запущен python3, на экране будет символ >>> в начале строки, где вы можете что-то набирать. Это и называется командной строкой интерпретатора Python 3.x (рис. 86). Попробуем напечатать в окне приглашения print('Привет') и нажать клавишу Enter. В результате должно появиться слово «Привет». Как видим, синтаксис языка подсвечивается, результат работы программы выделяется синим цветом.

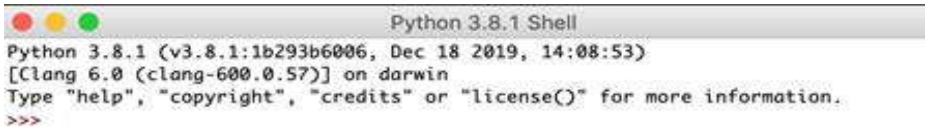

Рис. 86. Окно IDLE (Python GUI)

Рис. 87. Окно IDLE (Python GUI)

Стоит также заметить, что Python 3.x выдаёт результат работы строки немедленно. Но не всегда удобно с точки зрения работы с разными программами набирать их в команд-

ной строке интерпретатора. Очень важной составляющей является сохранение программы в файл, чтобы потом можно было запускать её неограниченное количество раз.

На практике в современном программировании используются различные редакторы кода. Для удобного написания программ на Python 3 нам понадобится редактор кода для работы с файлами наших программ. Стоит очень ответственно подойти к выбору редактора кода. Тут нужно отметить, что хороший редактор поможет вам легко писать программы на Python 3, сделает это понятно, быстро, комфортно, поможет быстрее реализовать свои идеи.

В настоящее время в кругу разработчиков чётко определились основные требования к редактору, такие как: умный поиск, подсветка синтаксиса, обработчик ошибок, поддержка баз данных и т. д.

IDE (или интегрированная среда разработки) — программа, предназначенная для разработки программного обеспечения. Как следует из названия, IDE включает в себя инструменты, специально предназначенные для разработки программного обеспечения.

Стандартно инструменты содержат:

- редактор, предназначенный для обработки кода (например, с подсветкой синтаксиса и автозавершением);
- средства сборки, выполнения и отладки;
- систему контроля версий.

Если рассматривать популярные IDE, то они поддерживают множество языков программирования и содержат другие специализированные дополнительные функции. Данные продукты занимают внушительное место и, как правило, работают медленно.

Можно сформулировать требования к среде разработки на Python 3.x который упрощает разработку на данном языке программирования:

- Возможность сохранения и перезагрузки файлов. IDE или редактор позволят сохранить работу и открыть её позже, в том же состоянии, в котором она была до закрытия.
- Запустить код в среде программирования. С помощью IDE запустить код Python будет не сложнее, чем из простого текстового редактора.
- Поддержка отладки. Возможность проверить код вовремя перед запуском — особенность всех IDE и прочих редакторов кода.
- Подсветка синтаксиса. Определение ключевых слов, переменных и символов в коде делает чтение и понимание кода намного проще.
- Автоматическое форматирование кода. Любой редактор или IDE будет распознавать двоеточие в конце for или while.

Есть много других функций: управление исходным кодом, модель расширения, инструменты сборки и тестирования, помощь с синтаксисом языка и др. Но приведённый выше список — основные функции, которые поддерживает хорошая среда редактирования.

Учитывая эти особенности, давайте рассмотрим некоторые инструменты общего назначения, которые используют для разработки в Python.

Выделяют два вида:

1) среда разработки с поддержкой языка Python;
2) специализированные редакторы и IDE для Python.

К первому виду можно отнести такие, как Sublime Text, ATOM, GNU Emacs, Vim, Eclipse, Visual Studio Code.

Sublime Text — редактор кода, который поддерживается на всех современных платформах, в том числе Windows, имеет встроенную поддержку редактирования кода Python и большой набор расширений (называемых пакетами), с помощью которых возможности синтаксиса и редактирования расширяются.

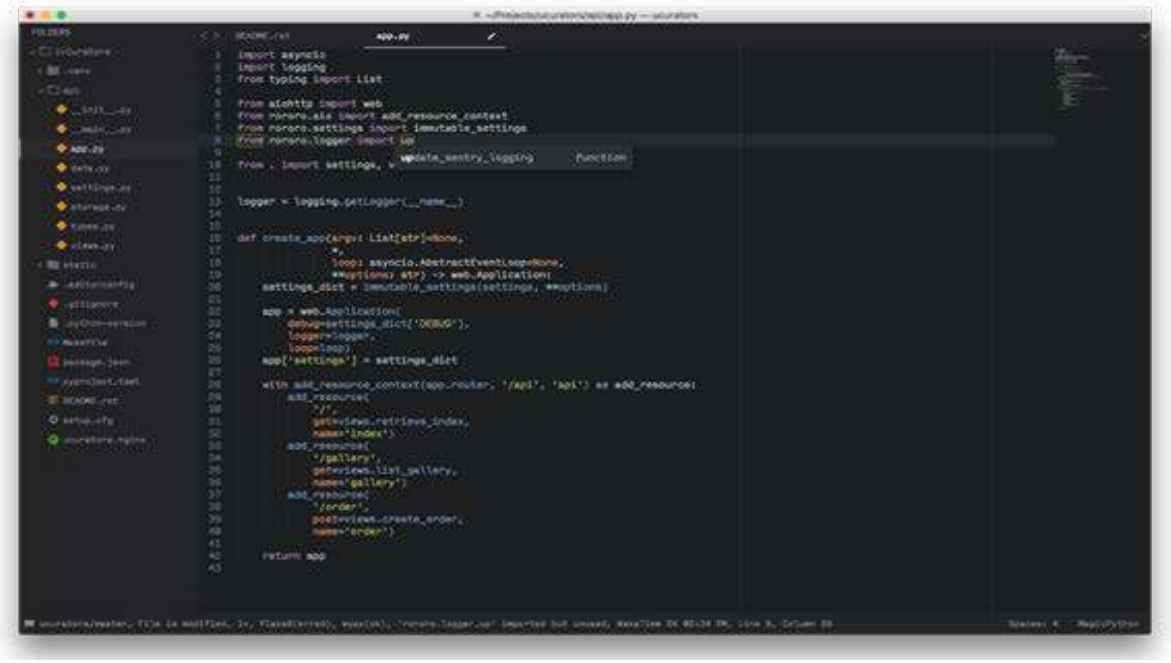

Рис. 88. Интерфейс программы Sublime Text

В таблице 11 отмечены основные достоинства и недостатки Sublime Text.

Таблица 11

Достоинства	Недостатки
Быстрая навигация, командная палитра, API плагинов на Python, одновременное редактирование, высокая степень настраиваемости. Поддерживает много языков программирования. Считается достаточно лёгким и быстрым редактором кода	Пакеты Sublime Text написаны на Python, и для установки редактора часто требуется выполнить скрипты Python непосредственно в Sublime Text. Программа не является бесплатной, есть период пробной версии. Установка расширений требует дополнительных усилий. В редакторе отсутствует прямая поддержка для выполнения или отладки кода из редактора

Atom — бесплатный текстовый редактор с открытым исходным кодом для Windows с поддержкой плагинов, написанных на JavaScript. Редактор основан на Electron (ранее известный как Atom Shell) — фреймворке кросс-платформенной разработки с использованием Chromium и io.js. Поддержка языка Python обеспечивается расширением, которое можно установить при запуске Atom (рисунок 88).

В таблице 12 отмечены основные достоинства и недостатки Atom.

Таблица 12

Достоинства	Недостатки
Достаточно лёгкий и быстрый редактор. Поддерживает много языков программирования, каждый язык работает при наличии предварительно установленного компилятора/ интерпритатора. Включает в себя отладчик, инструменты для работы с Git, подсветку синтаксиса, дополнительно подключаемые плагины	Редактор построен на Electron, это означает что он работает как процесс JavaScript, а не как отдельное приложение

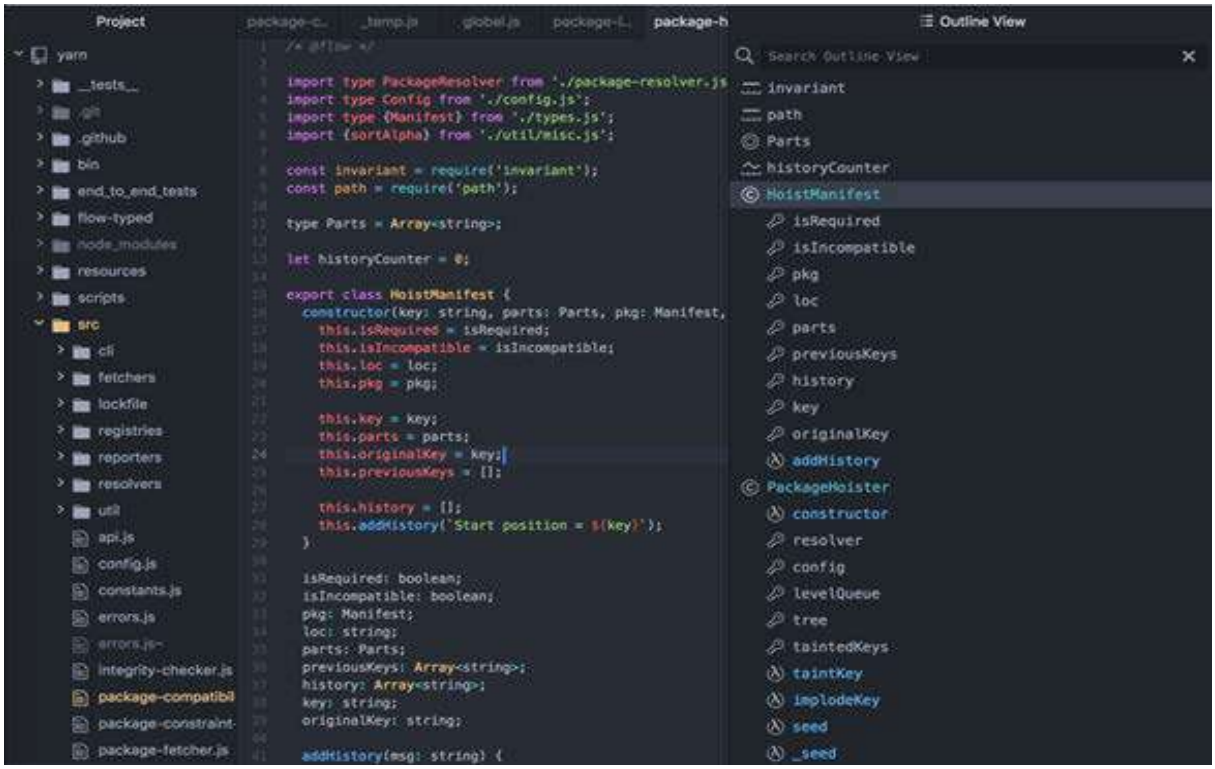

Рис. 89. Интерфейс программы Atom

Visual Studio Code — полнофункциональный редактор кода, доступный для различных платформ, в том числе и Windows. Представляет собой достаточно универсальный, быстрый, полнофункциональный редактор кода (VS-Code) с открытым исходным кодом, масштабируемый и настраивается под большинство задач. Можно отметить тот факт, что, как и редактор кода Atom, VS-Code построен на Electron (рис. 90).

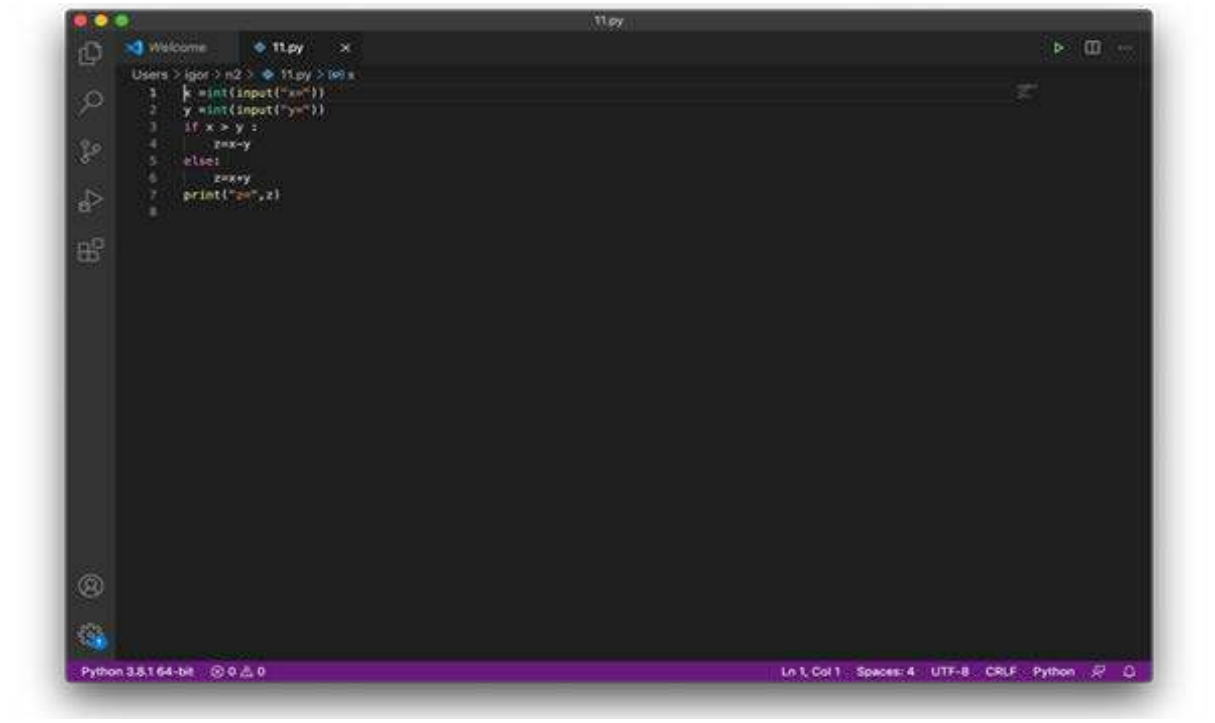

Рис. 90. Интерфейс программы Visual Studio Code

В таблице 13 отмечены основные достоинства и недостатки Visual Studio Code.

Таблица 13

Достоинства	Недостатки
Достаточно простой и лёгкий редактор кода для кросс-платформенной разработки веб-приложений. Содержит отладчик, инструменты для работы с Git, подсветку синтаксиса, дополнительно подключаемые плагины, средства для рефакторинга. В программе хорошо представлены возможности для кастомизации: пользовательские темы, сочетания клавиш и файлы конфигурации. Доступен для Windows, macOS и Linux	Редактор построен на Electron, это означает что он работает как процесс JavaScript, а не как отдельное приложение

Второй вид программ — это специализированные редакторы и IDE для Python 3.x.

1. IDE PyCharm компании Jetbrains. Одна из лучших полнофункциональных выделенных IDE для Python (рис. 91).

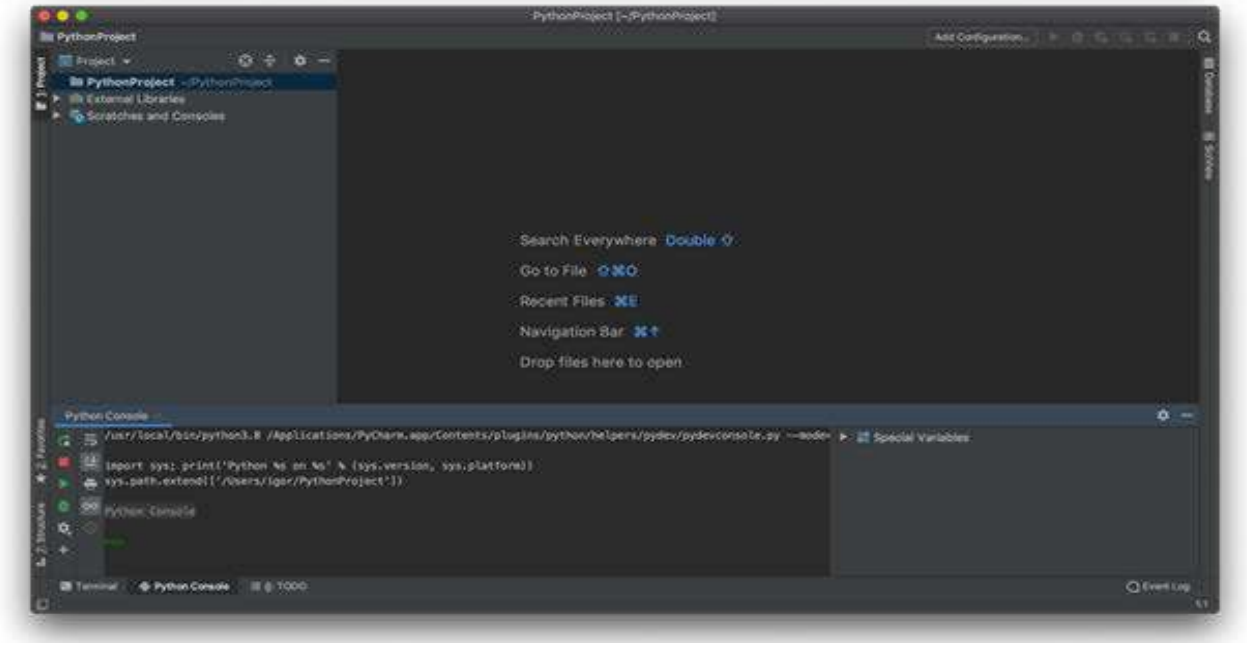

Рис. 91. Интерфейс программного продукта IDE PyCharm

В таблице 14. отмечены основные достоинства и недостатки IDE PyCharm.

Таблица 14

Достоинства	Недостатки
Все возможности IDE. Управление версиями и проектами, нативный запуск и написание кода, подсветка синтаксиса, автозавершением кода Python и т. д. IDE доступна бесплатная версия с открытым исходным кодом (Community) для Windows, macOS и Linux	Медленная загрузка среды разработки, требует более углубленную настройку для проектов

2. IDE Spyder — это программа с открытым исходным кодом, оптимизированная для области анализа данных. Spyder поставляется с дистрибутивом диспетчера пакетов Anaconda, поэтому зависит от уже установленных настроек (рис. 92).

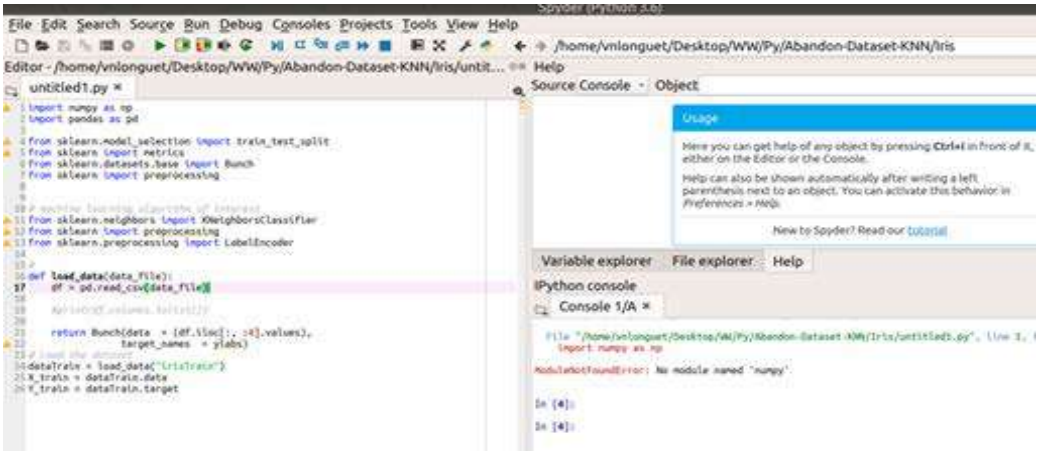

Рис. 92. Интерфейс программного продукта IDE Spyder

В таблице 15 отмечены основные достоинства и недостатки IDE PyCharm.

Таблица 15

Достоинства	Недостатки
Хорошая интеграция с библиотеками для анализа данных в Python, такими как SciPy, NumPy и Matplotlib, IPython, Jupyter. Проводник переменных Spyder, который отображает данные с использованием табличного макета внутри среды разработки. Обладает возможностями современной IDE: редактор кода с надёжной подсветкой синтаксиса, автозавершением кода Python и встроенный браузер с документацией. IDE доступна бесплатно для Windows, macOS и Linux	Не поддерживает расширенный функционал для опытных разработчиков, в большей степени направлен на узкую область по анализу данных

3. IDE Thonny. Данная интегрированная среда разработки для языка Python в первую очередь направлена на начинающих разработчиков (рис. 93).

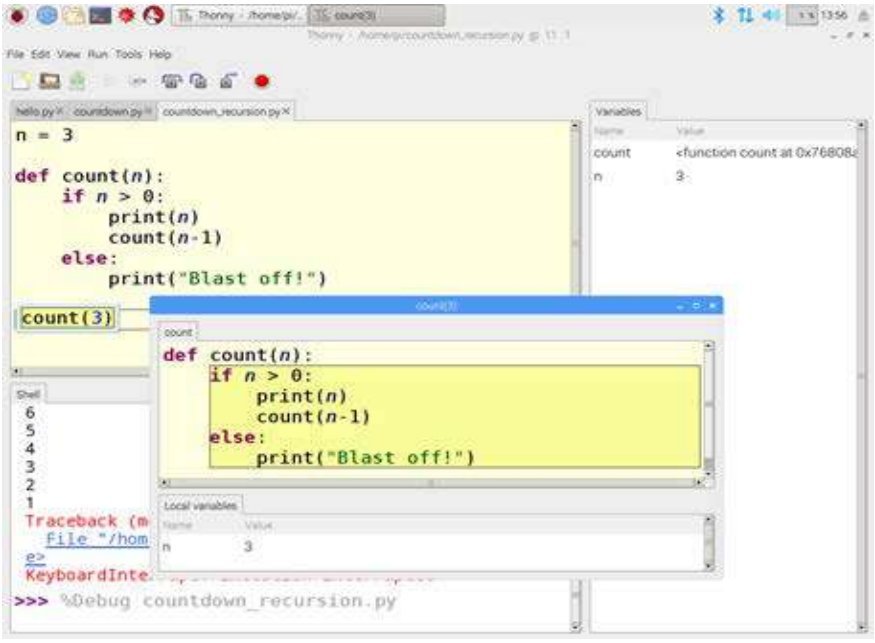

Рис. 93. Интерфейс программного продукта IDE Thonny

Она поддерживает различные способы пошагового выполнения кода, пошаговое вычисление выражений, детальную визуализацию стека вызовов и режим объяснения концепций ссылок. Программа работает на всех известных ОС, в том числе Windows. Доступна в виде бинарного пакета, включающего актуальную версию интерпретатора Python. Также IDE Thonny можно установить через менеджер пакетов.

В таблице 16 отмечены основные достоинства и недостатки IDE Thonny.

Таблица 16

Достоинства	Недостатки
Поддержка номера строк, пошаговое выполнение выражений без точек останова, живые переменные во время отладки, поэтапное вычисление выражений, отдельные окна для выполнения вызовов функций, возможность регистрировать действия пользователя для воспроизведения или анализа процесса программирования	Не поддерживает расширенный функционал для опытных разработчиков, проект достаточно молодой, может содержать ошибки

Среда программирования Scratch

Планируемые результаты освоения учебного предмета с описанием универсальных учебных действий, достигаемых обучающимися

Познавательные действия:

— развитие алгоритмического и логического мышления;
— развитие умений постановки задачи, выделения основных объектов, математическое модели задачи;
— развитие умения поиска необходимой учебной информации;
— формирование представления об этапах решения задачи;
— формирование алгоритмического подхода к решению задач;
— формирование умения построения различных видов алгоритмов (линейных, разветвляющихся, циклических) для решения поставленных задач;
— формирование умения использовать инструменты среды Scratch для решения поставленных задач;
— формирование умения построения различных алгоритмов в среде Scratch для решения поставленных задач;
— формирование навыков работы со структурой алгоритма;
— формирование ключевых компетенций проектной и исследовательской деятельности;
— формирование мотивации к изучению программирования.

Регулятивные действия:

— формирование умения целеполагания;
— формирование умения прогнозировать свои действия и действия других участников группы;
— формирование умения самоконтроля и самокоррекции.

Личностные действия:

— формирование профессионального самоопределения;
— формирование уважительного отношения к интеллектуальному труду;
— формирование смыслообразования.

Коммуникативные действия:

— формирование умения работать индивидуально и в группе для решения поставленной задачи;
— формирование трудолюбия, упорства, желания добиваться поставленной цели;
— формирование информационной культуры.

Формы контроля

Во время проведения курса предполагается текущий, промежуточный и итоговый контроль.

Текущий контроль осуществляется регулярно во время проведения каждого лабораторного занятия, заключается в ответе учащихся на контрольные вопросы, демонстрации полученных скриптов в среде Scratch, фронтальных опросов учителем.

Также в тематическом планировании предполагается две промежуточные контрольные работы.

Контрольная работа для проверки полученных навыков по темам «Линейные алгоритмы», «Условные алгоритмы»

1. Написать следующую программу в среде Scratch: *Пройти 200 шагов, повернуть на 90 градусов по часовой стрелке, пройти ещё 100 шагов.*
2. Написать следующую программу в среде Scratch: *Пройти 100 шагов, повернуть против часовой стрелки на 90 градусов, пройти 50 шагов.*
3. Написать программу в среде Scratch, изображающую следующий рисунок.

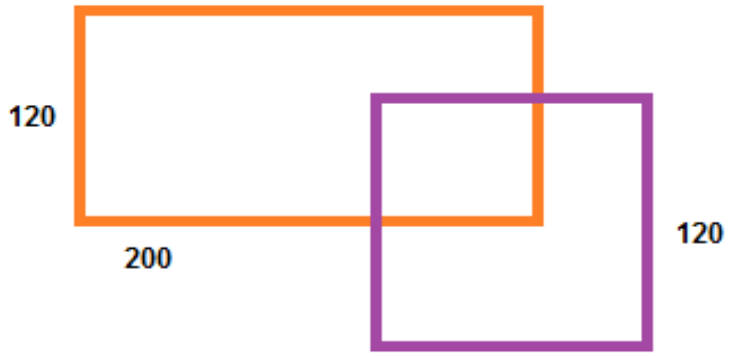

Рис. 94. Иллюстрация к задаче

4. Написать программу в среде Scratch, изображающую следующий рисунок.

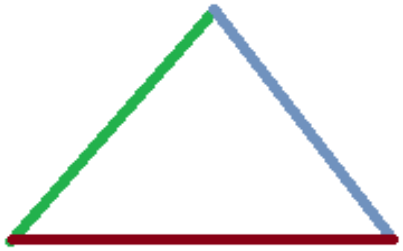

Рис. 95. Иллюстрация к задаче

5. Написать программу в среде Scratch, изображающую символику «Олимпийские кольца» (рис. 96).

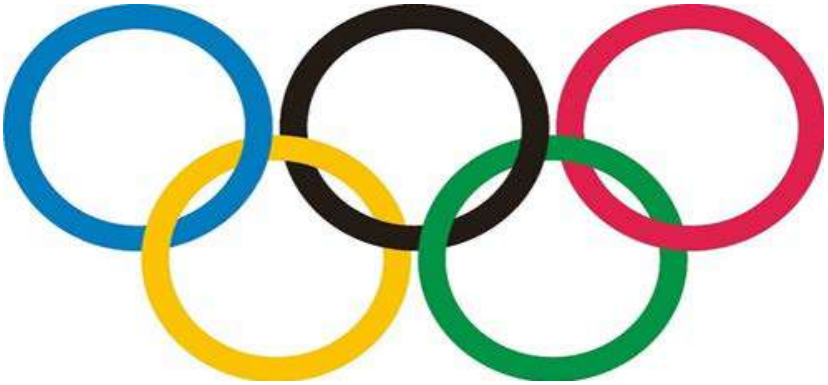

Рис. 96. Иллюстрация к задаче

6. Написать программу в среде Scratch: *Для введённых с клавиатуры чисел x и y вычислить значение выражения* $x^2 + y/3$.

7. Написать программу в среде Scratch: *Для введённых с клавиатуры чисел a и b выяснить, делится ли a на b.*

8. Написать программу в среде Scratch: *Пользователь вводит целое число. Программа должна ответить, чётным или нечётным является это число, делится ли оно на 3 и делится ли оно на 6.*

9. Написать программу в среде Scratch: *Пользователь вводит порядковый номер пальца руки (начиная с мизинца). Необходимо показать его название на экране.*

10. Написать программу в среде Scratch: *Пользователь вводит пароль. По данному паролю определите степени доступа: [0, 1000] — доступен модуль A, [1001, 2500] или [3000, 5000] — доступны модули B и C, [9400, 10000] или [10500, 50000] — доступен модуль D. Если значение не попало ни в один из указанных отрезков, то в доступе отказано!*

Контрольная работа для проверки полученных навыков по темам «Циклические алгоритмы», «Работа со списками»

1. Написать программу в среде Scratch, изображающую следующий рисунок.

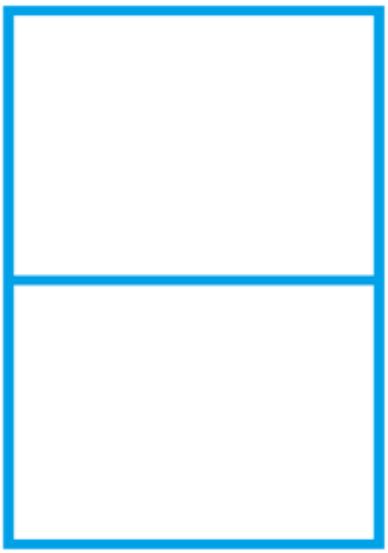

Рис. 97. Иллюстрация к задаче

2. Написать программу в среде Scratch, изображающую следующий рисунок.

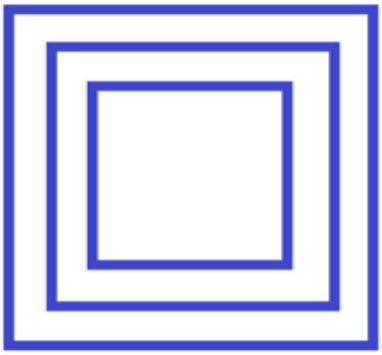

Рис. 98. Иллюстрация к задаче

3. Написать программу в среде Scratch, изображающую следующий рисунок.

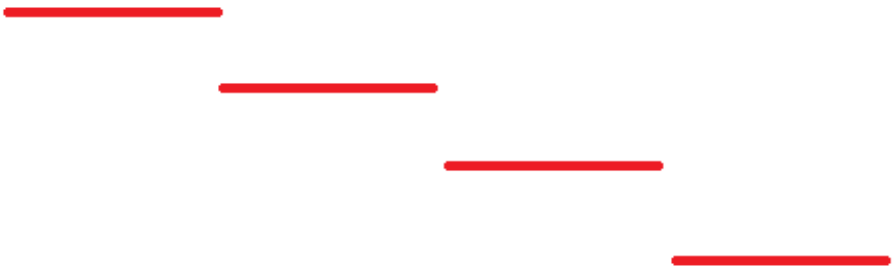

Рис. 99. Иллюстрация к задаче

4. Написать программу в среде Scratch: *Вывести на экран первые десять степеней двойки.*
5. Написать программу в среде Scratch*: Найти наибольший общий делитель двух чисел, введённых пользователем.*
6. Написать программу в среде Scratch*: В списке хранятся данные о температуре в городке N за 12 месяцев. Выведите температуру с марта по сентябрь.*
7. Написать программу в среде Scratch*: В списке хранятся данные о температуре в городке N за 12 месяцев. Выведите максимальную температуру за год.*
8. Написать программу в среде Scratch*: В списке хранится информация о четвертных оценках класса из 20 человек по информатике. Определите, сколько человек получили пятёрки за четверть.*
9. Написать программу в среде Scratch*: В списке хранится информация о четвертных оценках класса из 20 человек по информатике. Определите, какой процент хорошистов в классе.*
10. Написать программу в среде Scratch*: В списке хранится информация о четвертных оценках класса из 20 человек по информатике. Определите средний балл в классе.*

Также предполагается итоговая аттестация в форме разработки и защиты индивидуального проекта.

Планы учебных занятий

1. Знакомство со средой Scratch

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение навыков по работе в среде Scratch, освоение основных инструментов среды.

Метапредметные: способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные), делать выводы в процессе работы и по её окончании, корректировать намеченный план, ставить новые цели; умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, определять способы действий в рамках предложенных условий, корректировать свои действия в соответствии с изменяющейся ситуацией; умение оценивать правильность выполнения учебной задачи.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; сформированность их мотивации к обучению и целенаправленной познавательной деятельности.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: выполнение лабораторной работы 1.

2. Линейные алгоритмы

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты: получение навыков по работе с линейными алгоритмами в среде Scratch, освоение основных инструментов среды.

Метапредметные: способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные), делать выводы в процессе работы и по её окончании, корректировать намеченный план, ставить новые цели; умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, определять способы действий в рамках предложенных условий, корректировать свои действия в соответствии с изменяющейся ситуацией; умение оценивать правильность выполнения учебной задачи.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; сформированность их мотивации к обучению и целенаправленной познавательной деятельности.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: выполнение лабораторной работы 2.

3. Работа с переменными

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты: получение навыков по работе с переменными в среде Scratch, освоение основных инструментов среды.

Метапредметные: способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные), делать выводы в процессе работы и по её окончании, корректировать намеченный план, ставить новые цели; умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, определять способы действий в рамках предложенных условий, корректировать свои действия в соответствии с изменяющейся ситуацией; умение оценивать правильность выполнения учебной задачи.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; сформированность их мотивации к обучению и целенаправленной познавательной деятельности.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: выполнение лабораторной работы 3.

4. Условные алгоритмы

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты: получение навыков по работе с условными алгоритмами в среде Scratch, освоение основных инструментов среды.

Метапредметные: способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные), делать выводы в процессе работы и по её окончании, корректировать намеченный план, ставить новые цели; умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, определять способы действий в рамках предложенных условий, корректировать свои действия в соответствии с изменяющейся ситуацией; умение оценивать правильность выполнения учебной задачи.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; сформированность их мотивации к обучению и целенаправленной познавательной деятельности.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: выполнение лабораторной работы 4.

5. Циклические алгоритмы

Рекомендуемое количество часов на данную тему — 4 часа.

Планируемые результаты: получение навыков по работе с циклическими алгоритмами в среде Scratch, освоение основных инструментов среды.

Метапредметные: способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные), делать выводы в процессе работы и по её окончании, корректировать намеченный план, ставить новые цели; умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, определять способы действий в рамках предложенных условий, корректировать свои действия в соответствии с изменяющейся ситуацией; умение оценивать правильность выполнения учебной задачи.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; сформированность их мотивации к обучению и целенаправленной познавательной деятельности.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: выполнение лабораторной работы 5.

6. Создание подпрограмм

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты: получение навыков по работе со списками в среде Scratch, освоение основных инструментов среды.

Метапредметные: способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные), делать выводы в процессе работы и по её окончании, корректировать намеченный план, ставить новые цели; умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, определять способы действий в рамках предложенных условий, корректировать свои действия в соответствии с изменяющейся ситуацией; умение оценивать правильность выполнения учебной задачи.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; сформированность их мотивации к обучению и целенаправленной познавательной деятельности.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: выполнение лабораторной работы 6.

Описание лабораторных работ

Лабораторная работа 1. Знакомство со средой Scratch

Теоретическая часть

Как известно, программы пишутся на языке программирования. Программа же представляет собой набор инструкций, которые компьютер должен выполнить. Scratch представляет собой визуально-ориентированный язык программирования для детей.

Scratch был создан на языке Squeak, который представляет собой одну из разновидностей Smalltalk. Главным идеологом Scratch является ученик Пейперта Мич Резник из MIT Media Lab (Массачусетский технологический институт), именно в нём в 1968 г. С. Пейперт разработал Logo.

В настоящее время доступна онлайн-версия https://scratch.mit.edu/.

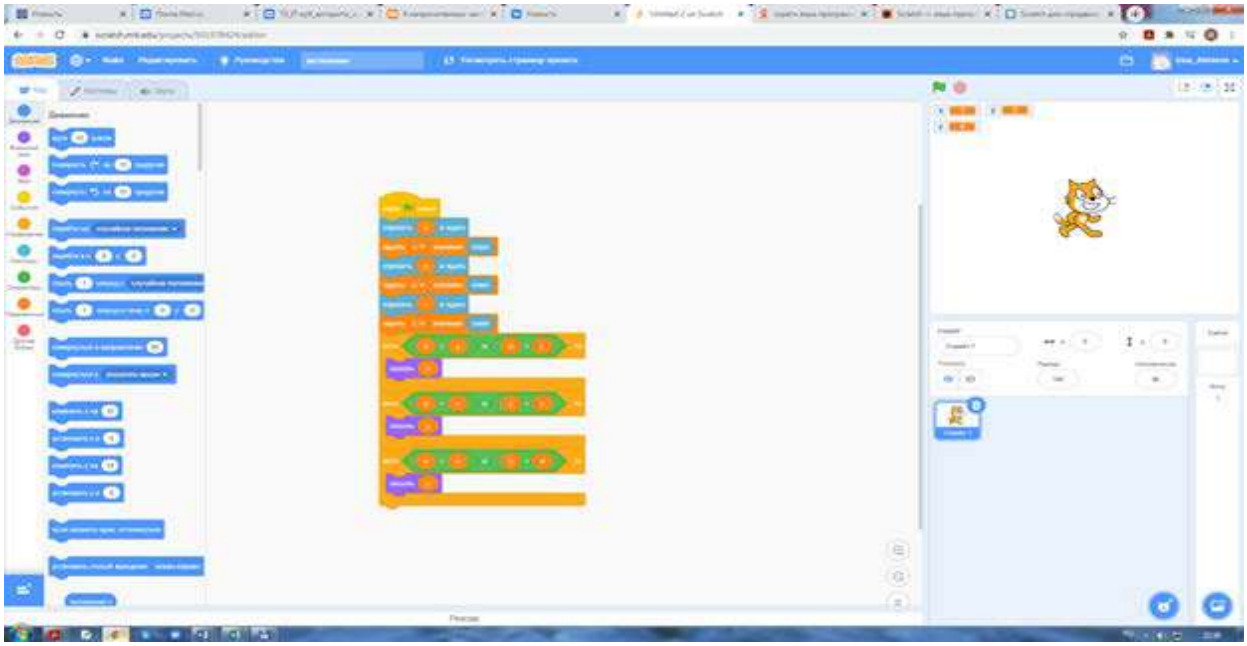

Рис. 100. Вид онлайн-среды Scratch

Scratch предназначен для визуального программирования. Программа на языке Scratch состоит из блоков, описывающих действия, которые выполняет спрайт. Роль спрайта по умолчанию выполняет рыжий кот, но есть возможность смены спрайта.

Подробное описание среды представлено в дидактических материалах.

Практическая часть

Цель работы: ознакомление со средой Scratch, изучение основных инструментов среды.

Ход лабораторной работы

1. Откройте среду Scratch.
2. Выберите спрайт для работы, например медведя (рис. 101).
3. Выберите фон из коллекции фонов, например, лес (рис. 102).
4. Смените спрайт, удалите фон.

Рис. 101. Вид спрайта Медведь

Рис. 102. Вид фона Лес

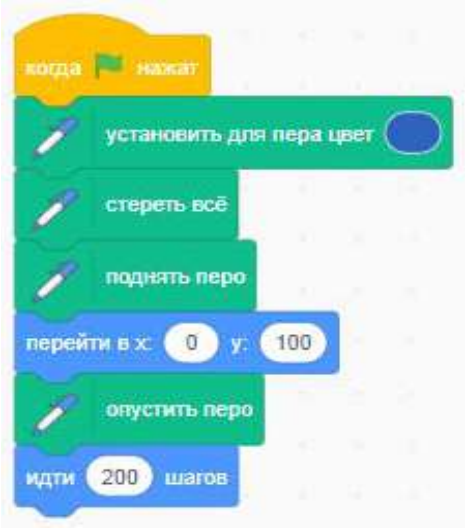

Рис. 103. Вид скрипта

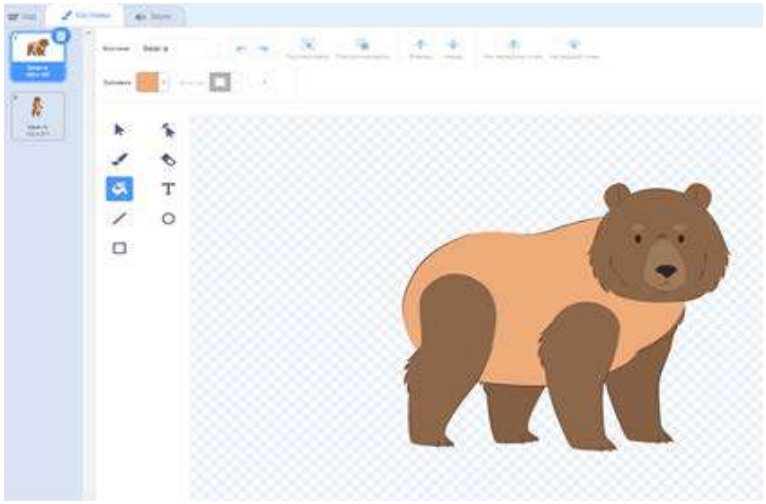

Рис. 104. Изменение спрайта в графическом редакторе

5. Выполните следующий скрипт (рис. 103).
6. Объясните, что выполняет скрипт.
7. Измените спрайт в графическом редакторе Scratch (рис. 104).
8. Выполните следующий скрипт (рис. 105).
9. Объясните, что выполняет скрипт.
10. Выполните следующий скрипт (рис. 106).
11. Объясните, что выполняет скрипт.

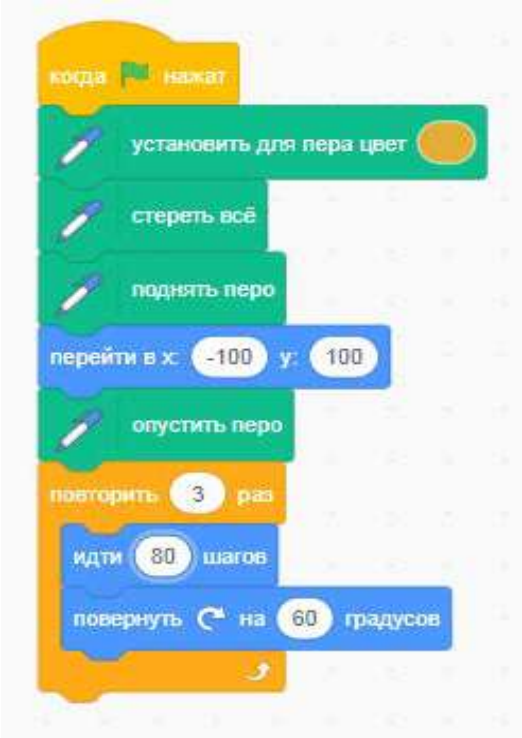

Рис. 105. Вид скрипта

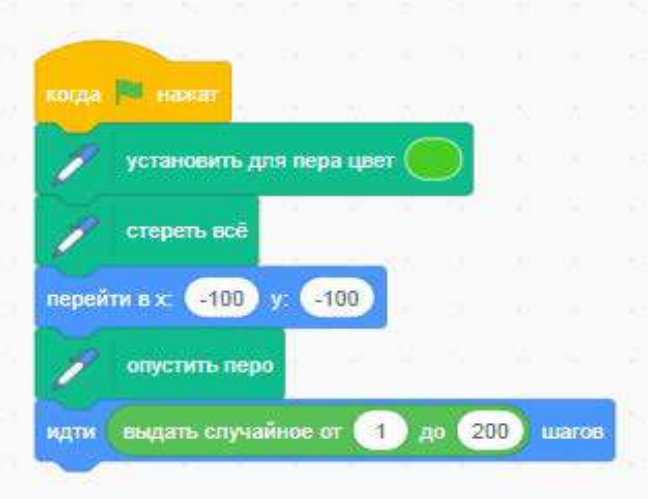

Рис. 106. Вид скрипта

Выводы: в процессе выполнения лабораторной работы вы получили представление о работе в среде Scratch.

Контрольные вопросы:

1. Как изменить вид спрайта в среде Scratch?
2. Можно ли редактировать фон в среде Scratch?
3. Какой вид графики можно создавать с помощью графического редактора в среде Scratch?

Лабораторная работа 2.
Линейные алгоритмы

Теоретическая часть

Линейный алгоритм — это вид алгоритма, который образует команды, выполняемые однократно, одна за другой, в той последовательности, в которой они были изначально описаны. Линейная структура одна из самых простых. Примерная блок-схема линейного алгоритма представлена на рисунке 107.

Линейные алгоритмы реализуются в любом языке программирования, можно привести множество примеров линейных алгоритмов, например алгоритм вычисления произведения двух целых чисел (рис. 108).

Рис. 107. Блок-схема линейного алгоритма

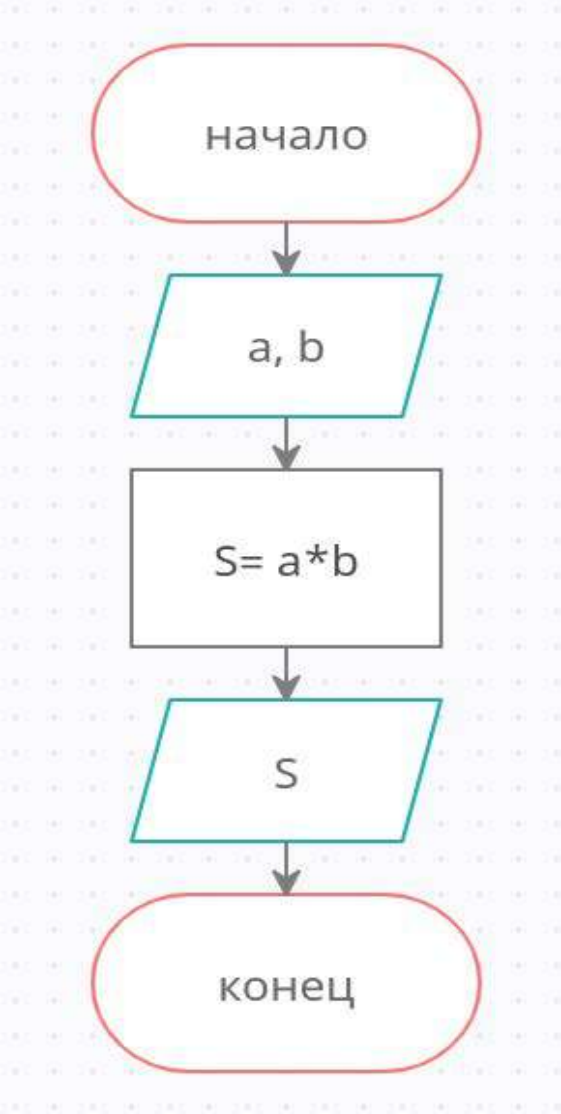

Рис. 108. Пример линейного алгоритма

Практическая часть

Цель работы: ознакомление с построением и выполнением линейных алгоритмов, работа с основными блоками в среде Scratch.

Ход лабораторной работы

Выполните в среде Scratch решение следующих задач.

1. Составьте программу «Пройти 10 шагов».
2. Составьте программу «Пройти 100 шагов, повернуться по часовой стрелке на 60 градусов».
3. Выполните следующую программу (рис. 109). Представляет ли она линейный алгоритм?

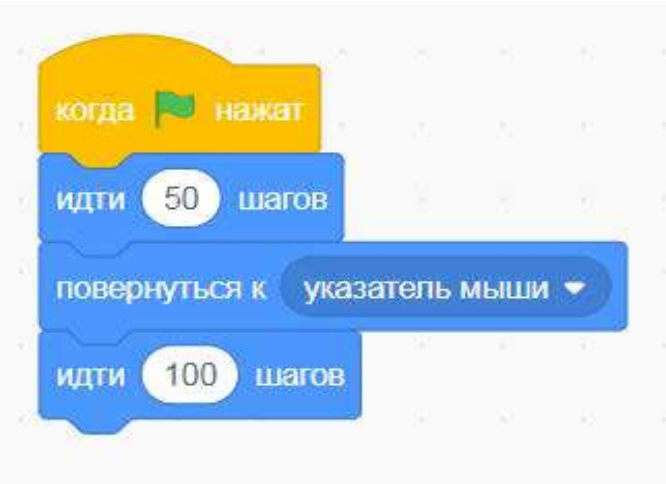

Рис. 109. Вид программы в среде Scratch

4. Пройдите 200 шагов, поверните на 45 градусов по часовой стрелке, пройдите ещё 200 шагов.
5. Поверните против часовой стрелки на 180 градусов, пройдите 200 шагов, поверните на 30 градусов по часовой стрелке, пройдите 100 шагов.
6. Установите зелёный цвет, пройдите 50 шагов, установите зелёный цвет, пройдите 10 шагов.

Выводы: в ходе выполнения лабораторной работы вы получили представление о создании линейных алгоритмов в среде Scratch.

Контрольные вопросы:

1. Дайте определение линейного алгоритма.
2. Приведите пример линейных алгоритмов.
3. Какие блоки вы использовали в лабораторной работе при создании линейных программ?

Лабораторная работа 3. Работа с переменными

Теоретическая часть

Говоря о работе с различными алгоритмами, конечно же нельзя пропустить работу с числами. Выполняя несложные линейные программы, которые обрабатывают числовые данные, например введённые пользователем, можно создать хороший задел на будущее,

подготавливая к составлению сложных алгоритмов как в среде Scratch, так и к написанию непосредственно программ в среде программирования.

Переменная представляет собой область памяти компьютера, которая имеет название и хранит внутри себя какие-либо данные.

Переменная в среде Scratch бывает простая (хранит одно значение), также может представлять собой список для хранения больше одного значения.

Для создания таких программ пользователю потребуется работа с переменными. Среда Scratch предоставляет нам возможность вводить в проект как простые переменные, так и массивы. Для этого необходимо обратиться к разделу «Переменные». Для создания переменной необходимо выбрать команду **Создать переменную** (рис. 110).

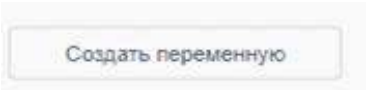

Рис. 110. Команда «Создание переменной»

В появившемся диалоговом окне указываем имя переменной.

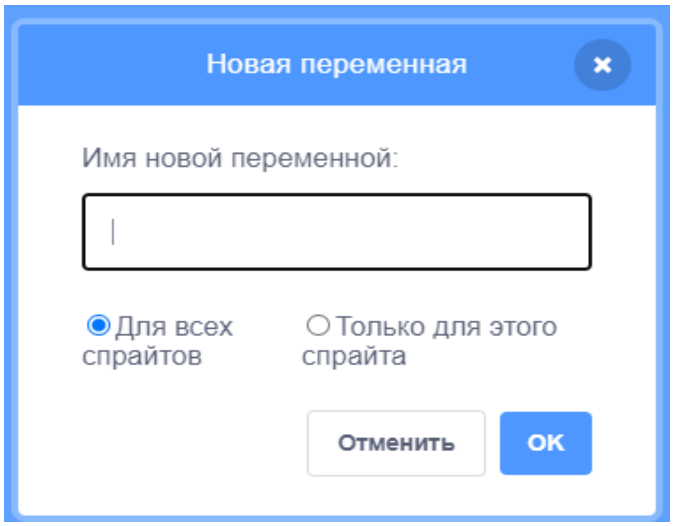

Рис. 111. Диалоговое окно для создания переменной

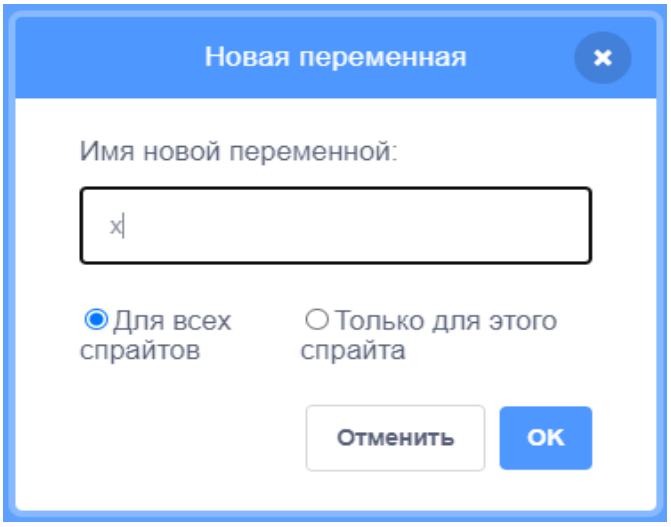

Рис. 112. Диалоговое окно для создания переменной

После задания имени переменной она появляется в проекте (рис. 113).

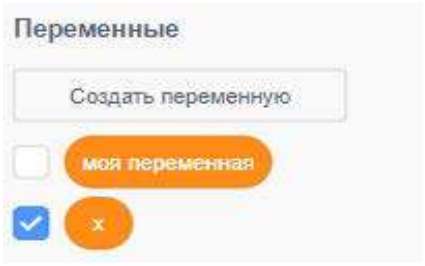

Рис. 113. Вид созданной переменной в разделе «Переменные»

Затем, после создания переменной, можно использовать различные команды для работы с ней, например задавать значение переменной, показывать значение переменной на сцене и т. д. Класс задач, решаемых с использованием переменных, очень широк.

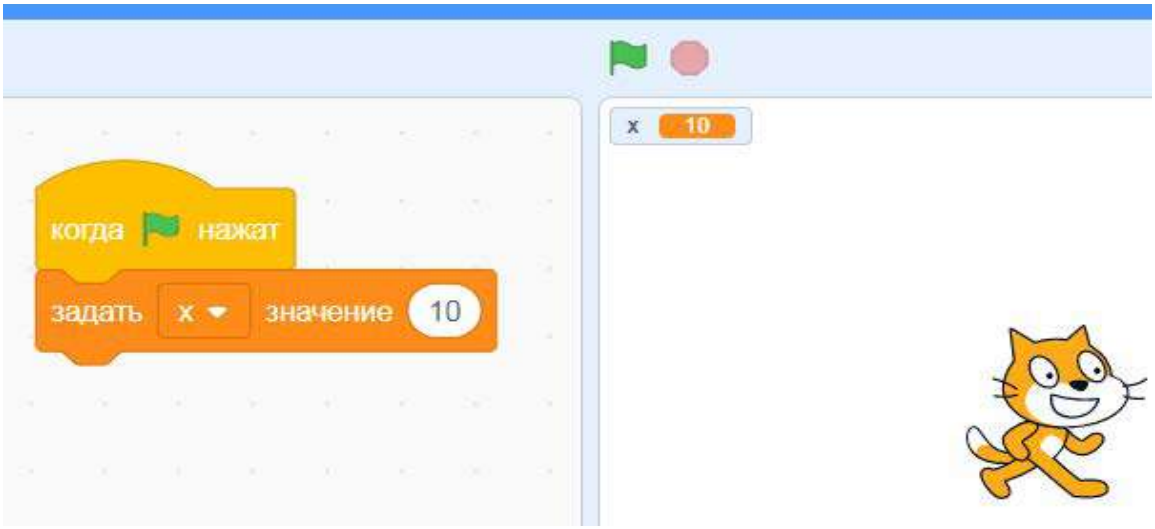

Рис. 114. Пример задания значения переменной

Опишем работу с основными блоками из раздела «Переменные».

Блок «Показать переменную...» используется для того, чтобы показывать значение переменной на сцене. Блок «Скрыть переменную...» используется, чтобы перестать показывать значение переменной на сцене (рис. 115). По умолчанию переменная и её значение отображается на сцене, но с помощью данных блоков мы можем скрывать или удалять переменную со сцены.

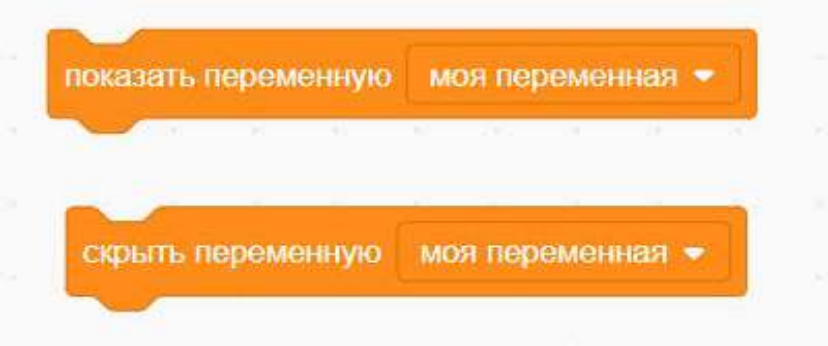

Рис. 115. Вид блоков по работе с переменными

Также можно изменять вид изображения переменной на сцене.

По умолчанию переменная показывается в левом верхнем углу сцены, но мы можем перемещать её в любое место с помощью мыши.

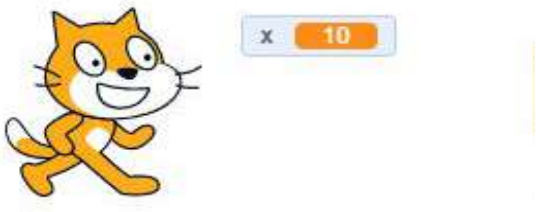

Рис. 116. Изменённый вид переменной на сцене

Практическая часть

Цель работы: ознакомление с основами работы с переменными в среде Scratch.

Ход лабораторной работы

1. Создайте в среде Scratch следующие переменные: x, c, fun. Какие инструменты среды вы использовали для этого?
2. Выполните следующую программу (рис. 117).
3. Выполните следующую программу (рис. 118).

Рис. 117. Вид программы

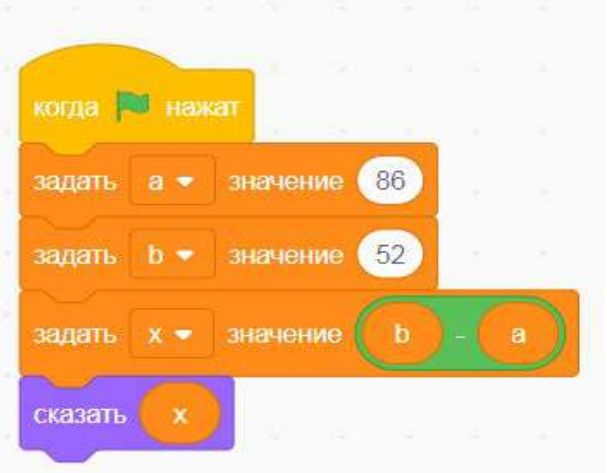

Рис. 118. Вид программы

4. Выполните в среде Scratch решение следующей задачи: «Ввести значение *x*. Прибавить к *x* число 15».
5. Выполните в среде Scratch решение следующей задачи: «Для введённых с клавиатуры чисел *c* и *d* вычислить значением выражения 3c — 2d».
6. Выполните в среде Scratch решение следующей задачи: «Написать алгоритм вычисления площади квадрата».

Выводы: в процессе выполнения лабораторной работы вы получили представление о работе с переменными в среде Scratch.

Контрольные вопросы:

1. Дайте определение **переменной**.
2. Какие виды переменных можно использовать в среде Scratch?
3. С помощью каких блоков можно задать значение переменной в среде Scratch?

Лабораторная работа 4.
Условные алгоритмы

Теоретическая часть

Условный алгоритм — это алгоритм, порядок выполнения действий в котором зависит от выполнения (или невыполнения) какого-либо условия.

Данный условный алгоритм, который также может носить название разветвляющийся, обеспечивает выбор одного из двух альтернативных путей в зависимости от истинности условия.

В этом случае логика принятия решения может быть записана следующим образом:

ЕСЛИ <условие> ТО <действия 1> ИНАЧЕ <действия 2> ВСЁ

Ветвление может быть реализовано в полной или в сокращённой форме (рис. 119).

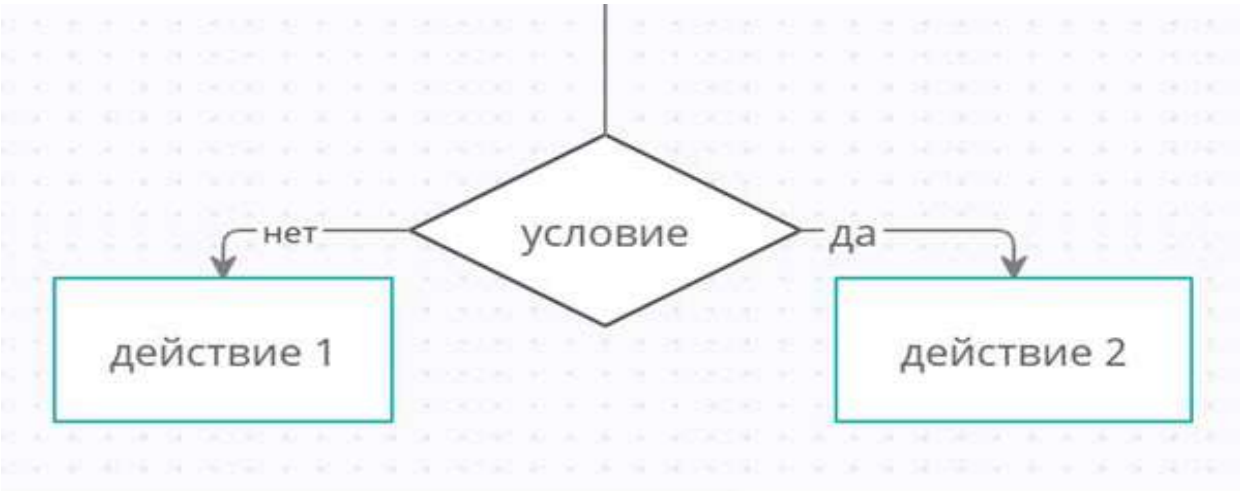

Рис. 119. Полная форма условного алгоритма

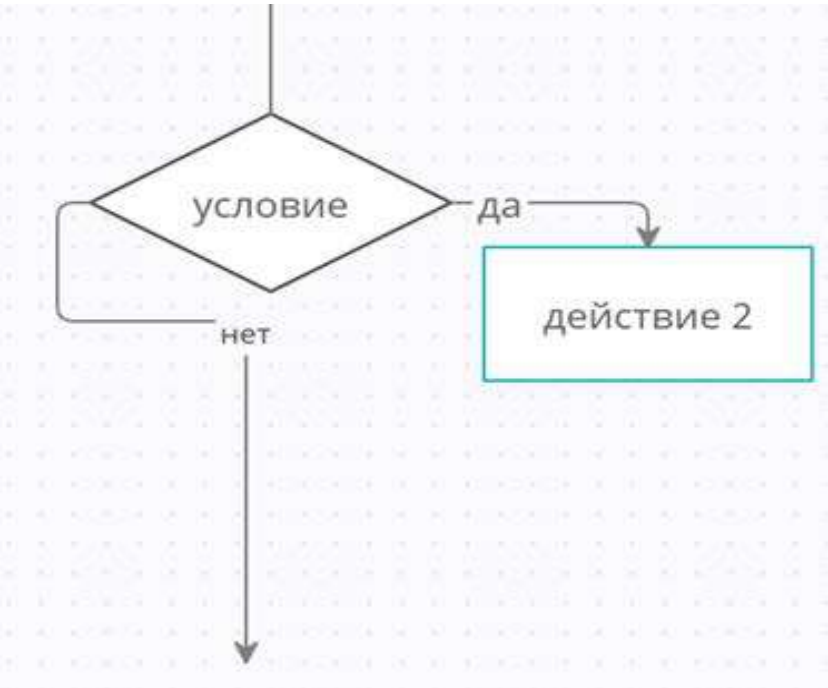

Рис. 120. Сокращённая форма условного алгоритма

Условные алгоритмы реализуются практически в любом языке программирования, не является исключением и Scratch. В среде Scratch для реализации разветвляющихся алгоритмов можно использовать два вида блоков, которые представлены в разделе «Управление».

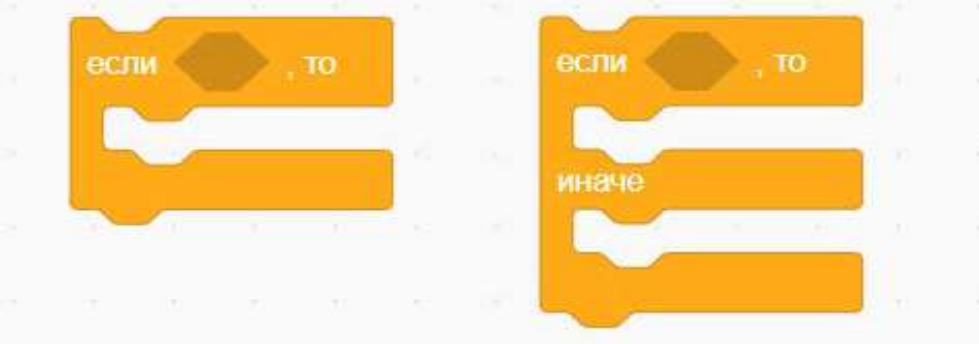

Рис. 121. Блоки для реализации сокращённой и полной формы ветвления

Практическая часть

Цель работы: ознакомление с основами работы с условными алгоритмами в среде Scratch.

Ход лабораторной работы

Создайте следующий скрипт (рис. 122). Объясните, что выполняет данный скрипт.

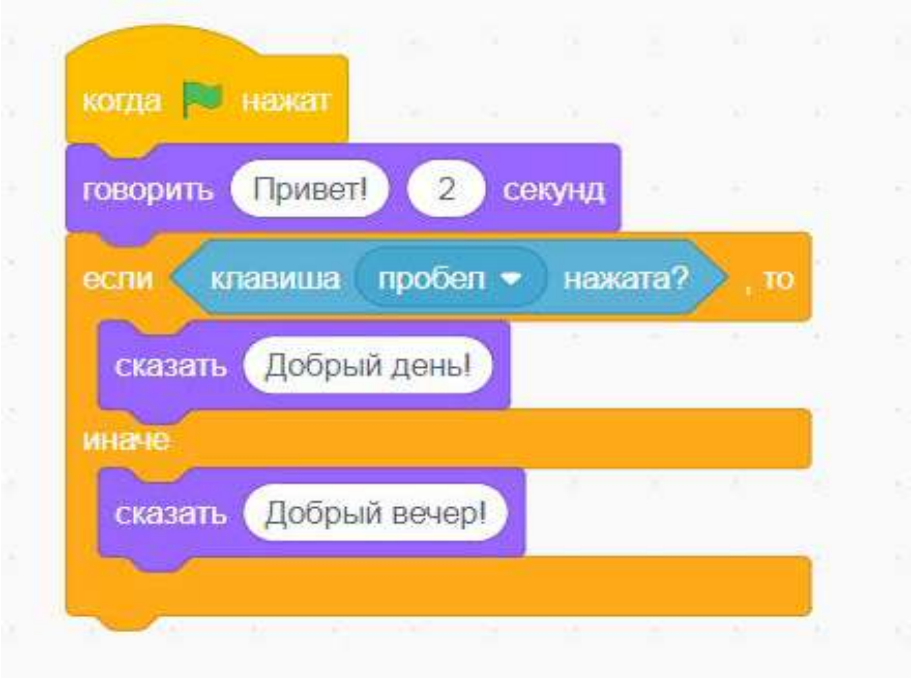

Рис. 122. Вид программы

Выполните в среде Scratch решение следующих задач:

1. Если введено число 5, то спрайт говорит пять, иначе спрайт говорит «неверно».
2. Найдите наибольшее из двух чисел, введённых пользователем.
3. Пройдите 100 шагов. Если спрайт коснулся края сцены, то оттолкнуться, если нет — идти ещё 200 шагов.

4. Запросите у пользователя ввод числа. Если введённое пользователем число чётное, то пройти 100 шагов, в противном случае пройдите 300 шагов.

Выводы: в процессе выполнения лабораторной работы вы получили представление о создании условных алгоритмов в среде Scratch.

Контрольные вопросы:

1. Дайте определение условного алгоритма.
2. Приведите пример условных алгоритмов.
3. Какие блоки используются в среде Scratch для реализации ветвлений?

Лабораторная работа 5. Циклические алгоритмы

Теоретическая часть

Операторы цикла используются для организации многократно повторяющихся вычислений. Любой цикл состоит из **тела** цикла, т. е. тех операторов, которые выполняются несколько раз, начальных установок, модификации параметра цикла и проверки условия продолжения выполнения цикла.

Один проход цикла называется **итерацией**. Проверка условия выполняется на каждой итерации либо до тела цикла (тогда говорят о цикле с предусловием), либо после тела цикла (цикл с постусловием). Разница между ними состоит в том, что тело цикла с постусловием всегда выполняется хотя бы один раз, после чего проверяется, надо ли его выполнять ещё раз. Проверка необходимости выполнения цикла с предусловием делается до тела цикла, поэтому возможно, что он не выполнится ни разу.

Для реализации циклических алгоритмов в среде Scratch используются несколько блоков. Далее рассмотрим каждый из них более подробно (рис. 123).

Данный блок используется для организации цикла с параметром, который повторяется заданное число раз. Внутри данного блока обычно располагаются несколько других блоков, которые и образуют тело цикла. Блок-схема работы данного блока представлена ниже (рис. 124):

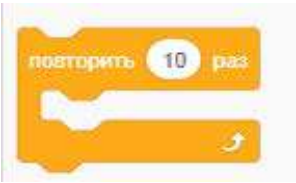

Рис. 123. Блок для организации цикла

Рис. 124. Блок-схема цикла

Рис. 125. Блок для организации цикла

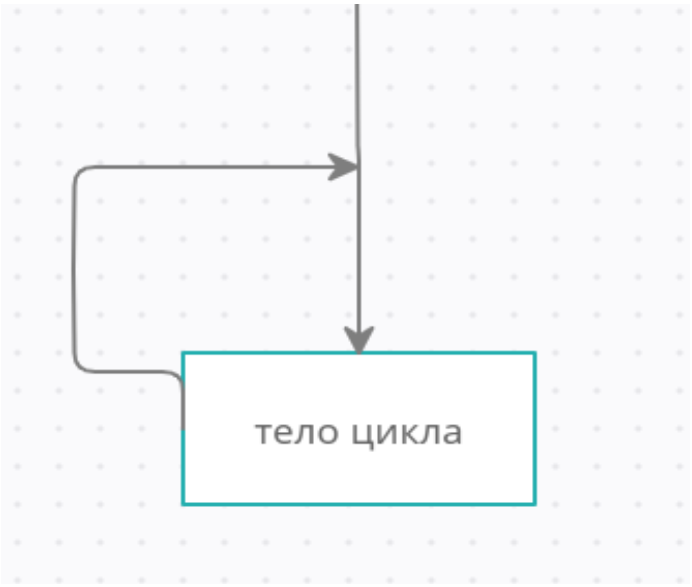

Рис. 126. Блок-схема цикла

Данный блок используется для организации бесконечных повторений. Блок-схема работы данного блока представлена ниже (рис. 126).

Также имеется блок для организации повторения тела цикла (блоков внутри), пока условие не станет истинным. Блок-схема работы данного блока представлена ниже (рис. 127):

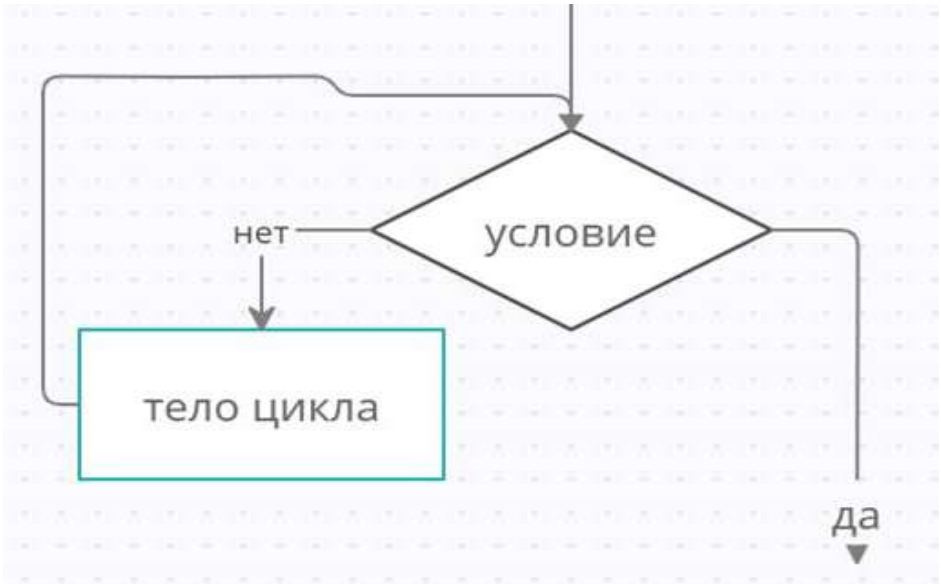

Рис. 127. Блок-схема цикла

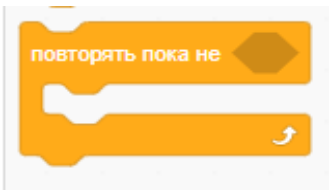

Рис. 128. Блок для организации цикла

Практическая часть

Цель работы: ознакомление с основами работы с циклическими алгоритмами в среде Scratch.

Ход лабораторной работы

1. Выполните следующую программу (рис. 129).

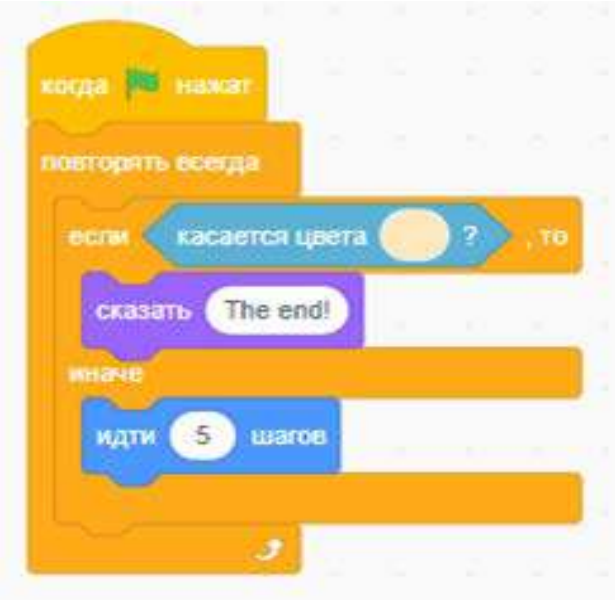

Рис. 129. Вид программы

2. Выполните следующую программу (рис. 130).

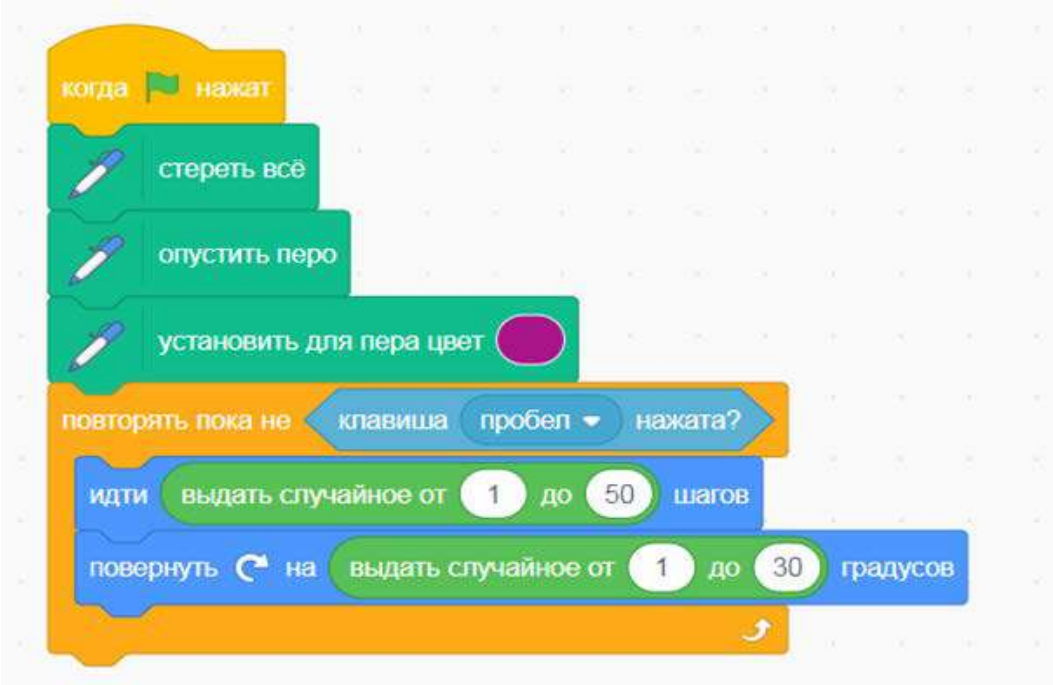

Рис. 130. Вид программы

Выполните в среде Scratch решение следующих задач.

3. Нарисуйте квадрат со стороной 100 (рис. 131).

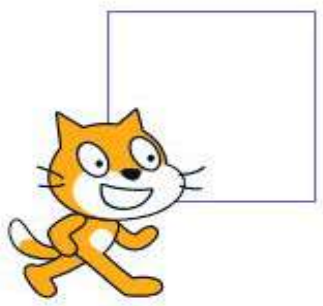

Рис. 131. Пример результат работы программы

4. Нарисуйте квадрат, величина стороны которого вводится пользователем.
5. Изобразите цифру 6 (рис. 132).

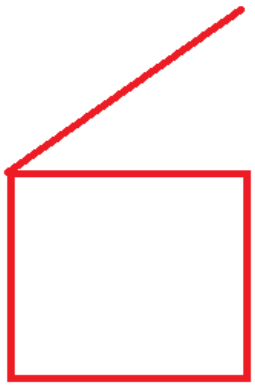

Рис. 132. Пример результат работы программы

Выводы: в процессе выполнения лабораторной работы вы получили представление о создании циклических алгоритмов в среде Scratch.

Контрольные вопросы:

1. Дайте определение **циклического алгоритма**.
2. Приведите примеры циклических алгоритмов.
3. Какие блоки используются для организации циклических алгоритмов в среде Scratch?

Лабораторная работа 6. Создание подпрограмм

Теоретическая часть

Подпрограммы являются средством реализации структурного программирования. Подпрограммы используются для следующих целей: первая — многократное использование в программе одного и того же фрагмента, а вторая — в случае, когда необходимо выполнить разделение сложной программы на составные части (процедурная декомпозиция). Также подпрограммы являются средством реализации восходящего и нисходящего проектирования.

В среде Scratch также имеются возможности создания и использования подпрограмм. Для этого необходимо обратиться к разделу «Другие блоки». В данном разделе пред-

ставлена всего одна команда — «Создать блок». При выборе данной команды появится диалоговое окно, представленное на рисунке 133.

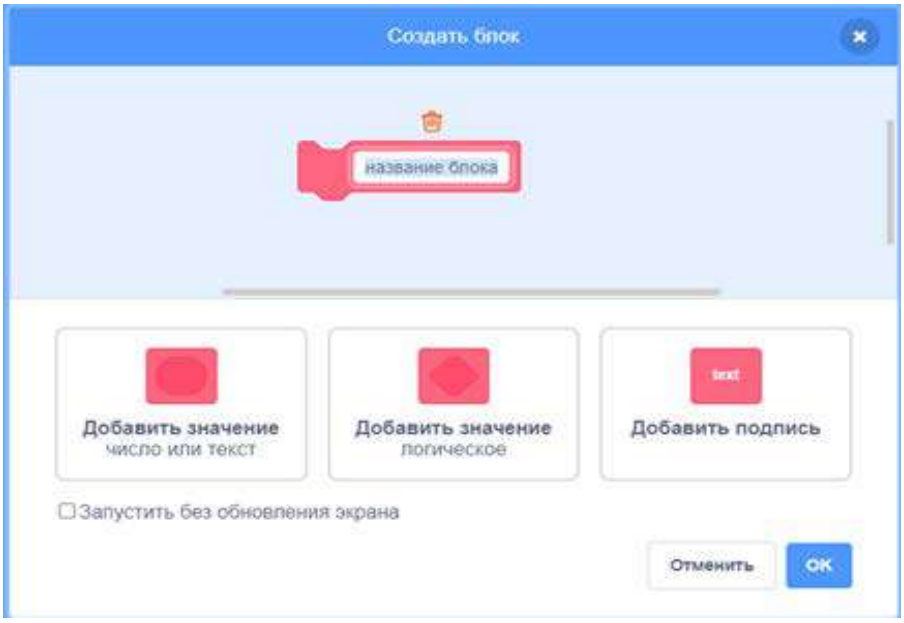

Рис. 133. Вид диалогового окна для создания подпрограммы (нового блока)

В этом окне необходимо ввести название нового блока, а также указать его параметры.

Блок можно создать и без параметров. Например, назовём его «треугольник». После того как вводится название нового блока, нажмем ОК, созданный блок появится в списке команд (рис. 134).

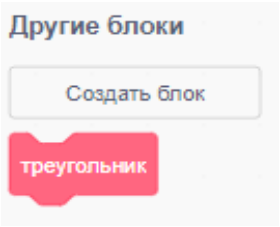

Рис. 134. Вид нового блока

Также в области для создания скриптов появится команда для определения скриптов нового блока (рис. 134).

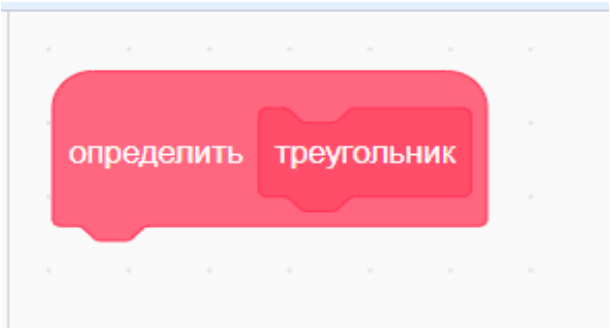

Рис. 135. Вид нового блока в области скриптов

Далее добавляем скрипт для рисования треугольника (рис. 136).

Рис. 136. Вид скрипта

Далее к этому блоку можно обращаться многократно для изображения треугольника. Создадим скрипт, который рисует один треугольник при нажатии клавиши **пробел** (рис. 137).

Рис. 137. Вид программы с новым блоком

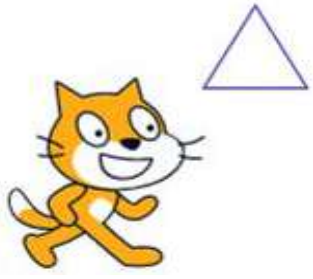

Рис. 138. Результат работы скрипта

Для этого можно объединить их изображение в новую подпрограмму (рис. 139).

Рис. 139. Вид программы с новым блоком

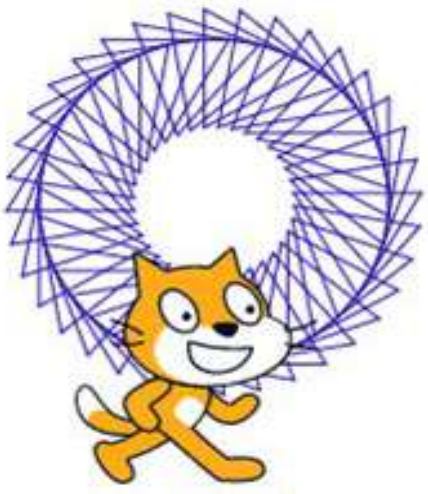

Рис. 140. Результат работы скрипта

Как видно на рисунке, скрипт изобразил несколько треугольников, объединённых в фигуру.

Как было сказано ранее, новый блок может иметь параметры. Параметры служат для передачи данных во внутрь подпрограммы. Разработаем новый блок, «треугольник1», у которого в качестве параметра будет размер треугольника. Также обратимся к команде «Создать блок». Но теперь в диалоговом окне необходимо также указать параметр — «Добавить значение число или текст» (рис. 141). При необходимости можно добавить и второй параметр, также выбрав соответствующую команду «Добавить значение».

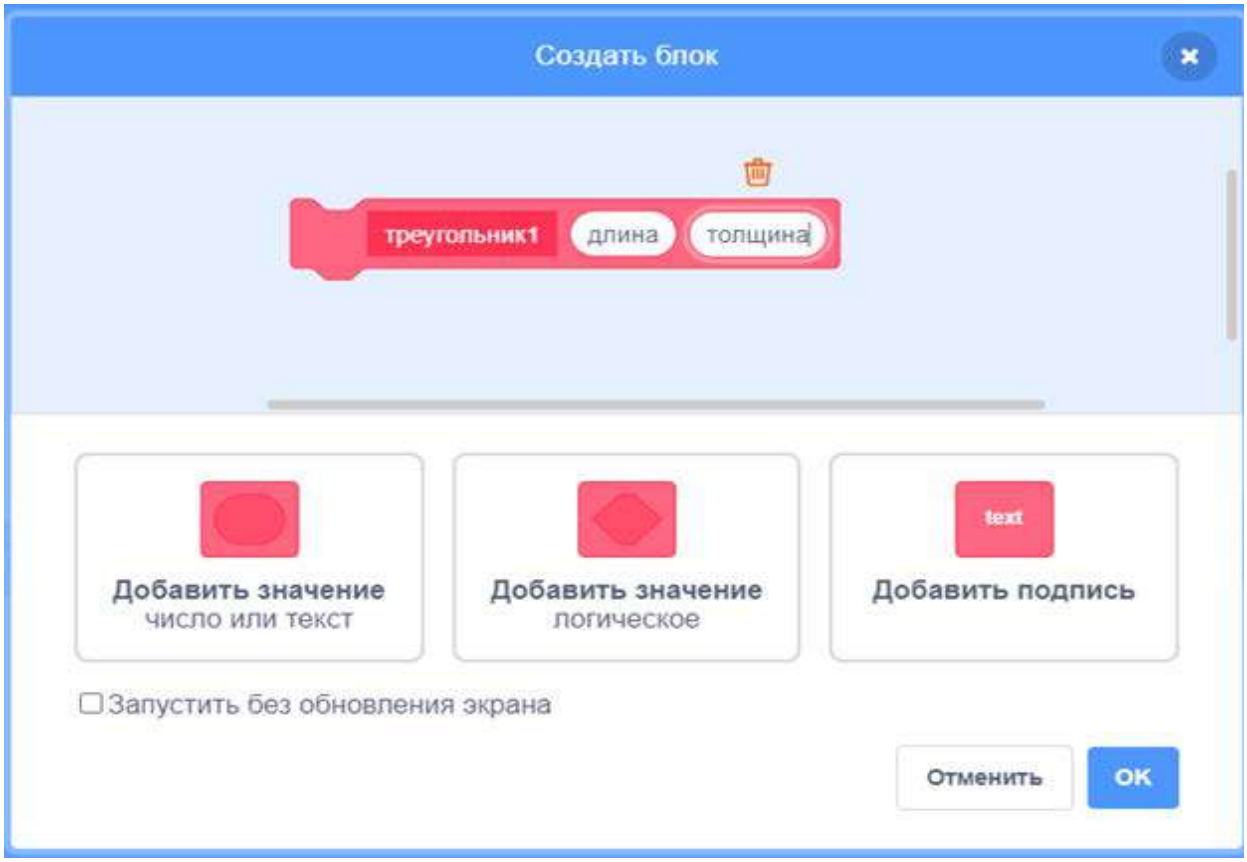

Рис. 141. Вид диалогового окна для создания подпрограммы с параметром

Далее создаём новый скрипт, используя параметры (рис. 142).

Рис. 142. Вид скрипта для создания нового блока

Как можно увидеть, в блоке включаются новые параметры, их можно просто перетаскивать мышью в необходимые пазы.

Далее создаётся скрипт, также изображающий два треугольника (рис. 143).

Рис. 143. Вид программы с новым блоком

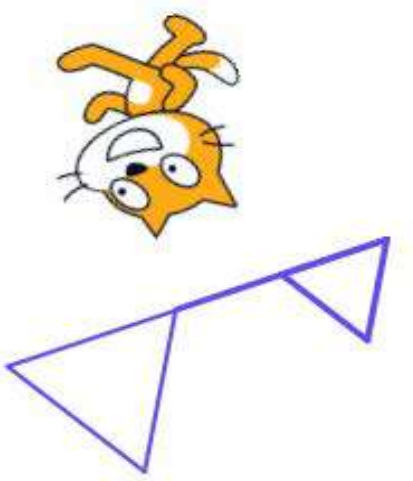

Рис. 144. Результат работы программы

Таким образом, можно создавать несколько новых блоков и использовать их в программе в среде Scratch.

Практическая часть

Цель работы: ознакомление с основами создания блоков-подпрограмм в среде Scratch.

Ход лабораторной работы

1. Создайте в среде Scratch новый блок, который реализует движение по сцене до края.
2. Откройте раздел «Другие блоки» и выберите команду «Создать блок».
3. Создайте блок без параметров и назовите его «Движение» (рис. 145).

Рис. 145. Создание нового блока

4. Для нового блока реализуйте следующий скрипт, представленный на рисунке 146.

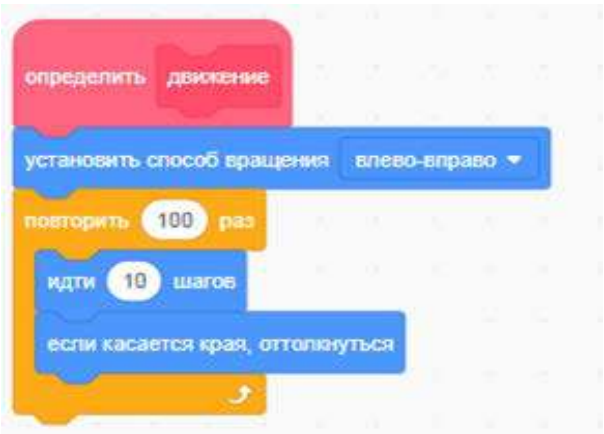

Рис. 146. Вид скрипта для нового блока

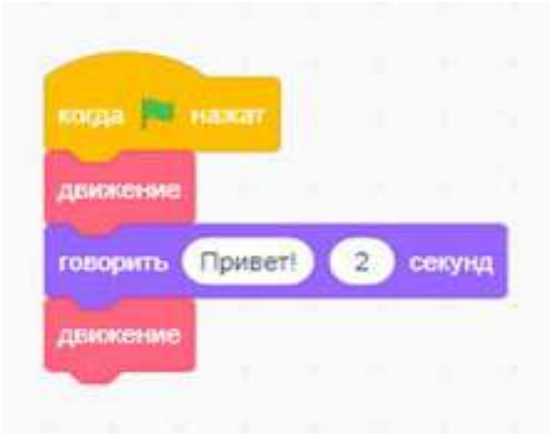

Рис. 147. Вид скрипта с новым блоком

5. Создайте новый скрипт, в котором используется блок «Движение» (рис. 147).
6. Создайте новый блок для изображения **квадрат**. Блок должен содержать параметр — размер стороны квадрата.

Выводы: в процессе выполнения лабораторной работы вы получили представление о создании новых блоков в среде Scratch.

Контрольные вопросы:

1. Дайте определение **подпрограммы**.
2. Для чего используются новые блоки в среде Scratch?
3. Для чего используются параметры при разработки новых блоков в среде Scratch?

Дидактические материалы

1. Описание разделов с командами

Можно дать краткую характеристику каждому разделу. Надо отметить, что практически все команды среды Scratch имеют описательное название, поэтому предполагают интуитивную реализацию.

Раздел **Движение** содержит группу команд, отвечающих за перемещение спрайта по сцене. Например, команда «идти *x* шагов» предназначена для перемещения спрайта на *x* шагов. Среди самых популярных также можно отметить команды: «повернуть на ... градусов», перейти в *x*:... *y*:...», «установить *x* в...», «установить *y* в ...» и т. д.

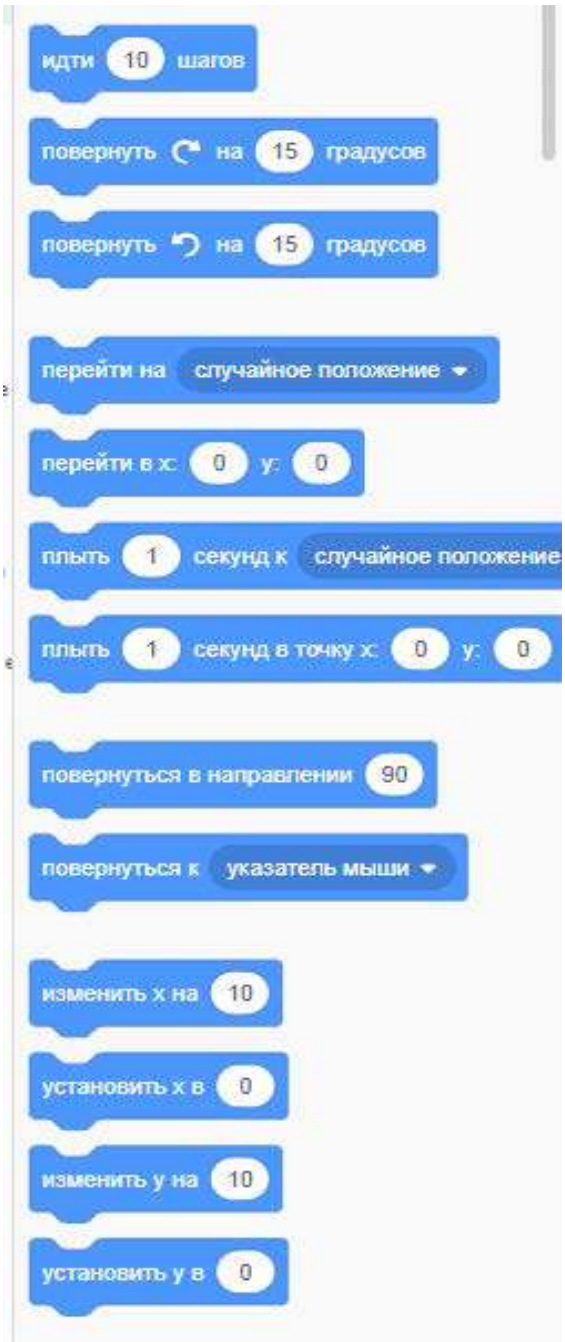

Рис. 148. Вид фрагмента раздела «Движение»

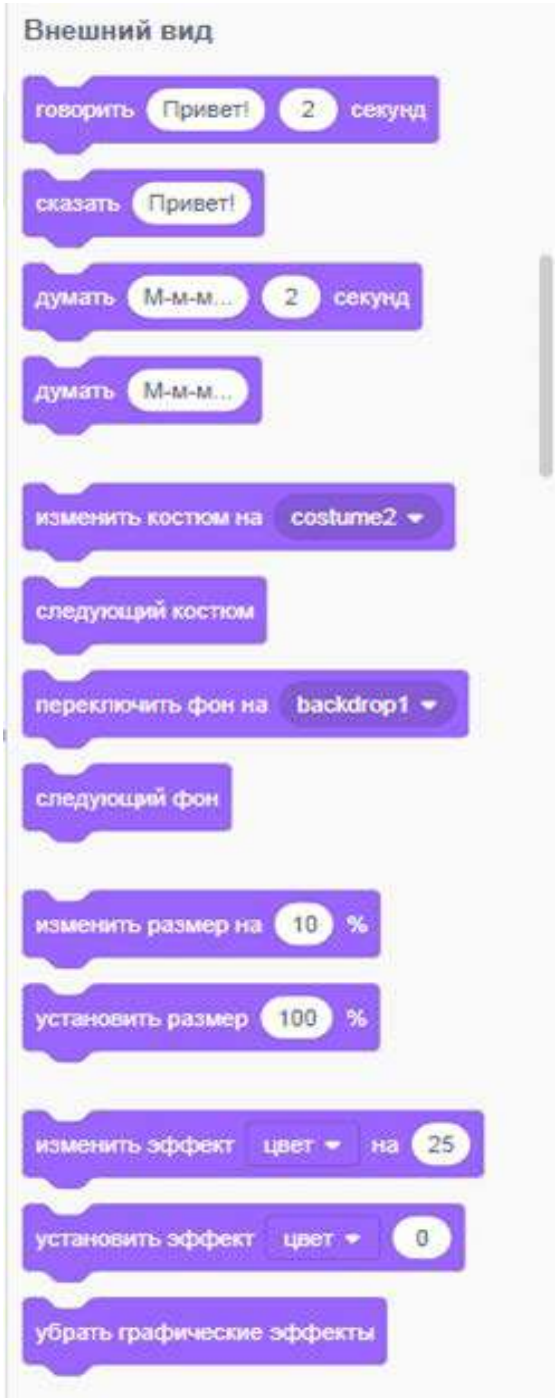

Рис. 149. Вид фрагмента раздела «Внешний вид»

Раздел **Внешний вид** предназначен для изменения внешнего вида спрайта. Например, изменение костюма спрайта, смена фона, видимость и невидимость спрайта и т. д. Команды данного блока могут использовать для вывода информации пользователю, например блок «сказать ...».

Раздел **Звук** предназначен для проигрывания звуков. Содержит команды выбора инструментов, изменения громкости звуков, темпа и т. д. Команды этого блока могут быть использованы для проигрывания музыкальных фрагментов (рис. 150).

Раздел **События** располагается в начале скрипта. Он позволяет скрипту реагировать на различные события, такие как нажатие клавиш на клавиатуре, щелчок мышью и т. д. В этом блоке сосредоточены в основном блоки-триггеры (рис. 151).

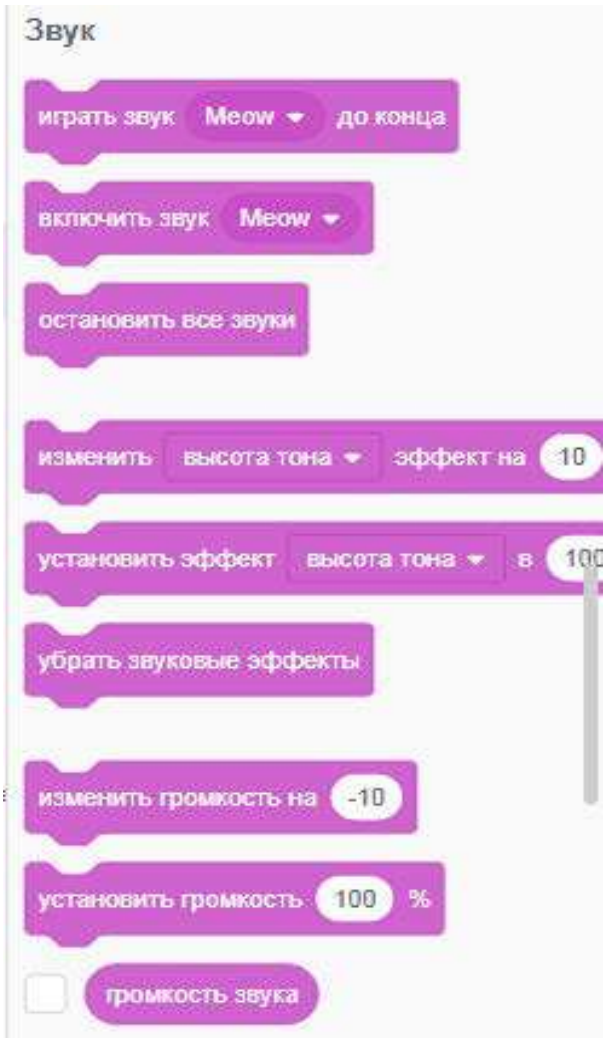

Рис. 150. Вид раздела «Звук»

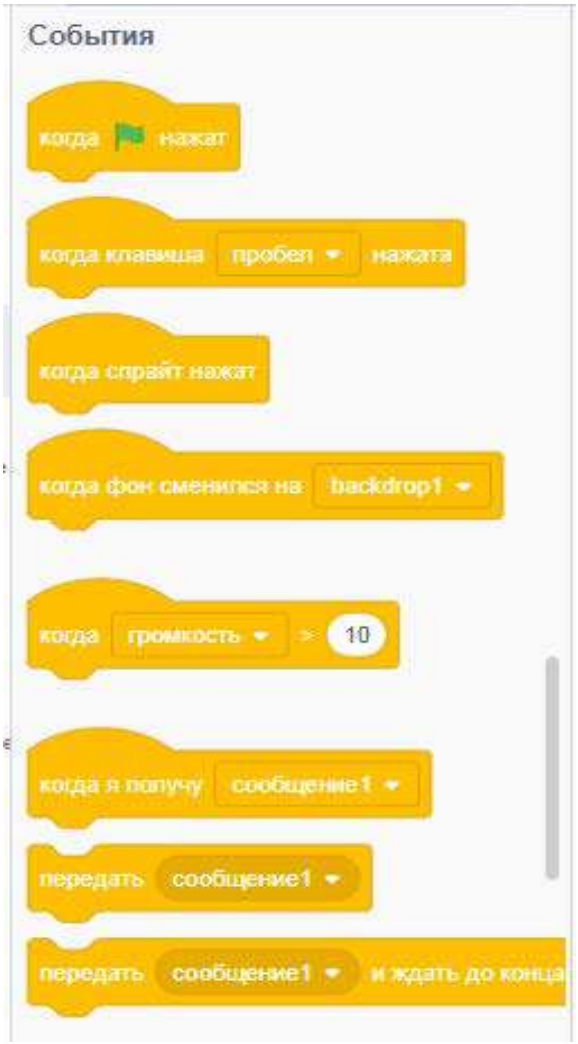

Рис. 151. Вид раздела «События»

Раздел **Управление** позволяет организовывать в программе такие конструкции, как ветвление, цикл различных видов. В данном разделе представлены как раз блоки управления (рис. 152).

Раздел **Переменные** отвечает за использование переменных в программе. Можно создавать, изменять значения как простых переменных, так и списков — фактически массивов (рис. 153).

Раздел **Сенсоры** содержит команды, позволяющие организовать ввод информации в компьютер. Например, работа с таймером, ввод информации пользователем и т. д. Также в состав данного раздела входят блоки-функции, которые возвращают, например, координату *x* спрайта (рис. 154).

Рис. 152. Вид фрагмента раздела «Управление»

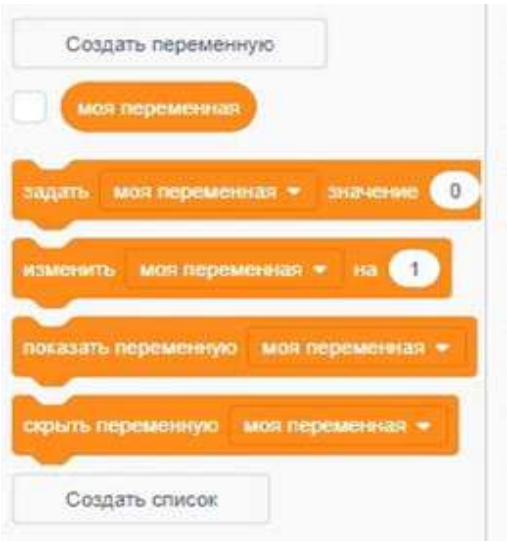

Рис. 153. Вид фрагмента раздела «Переменные»

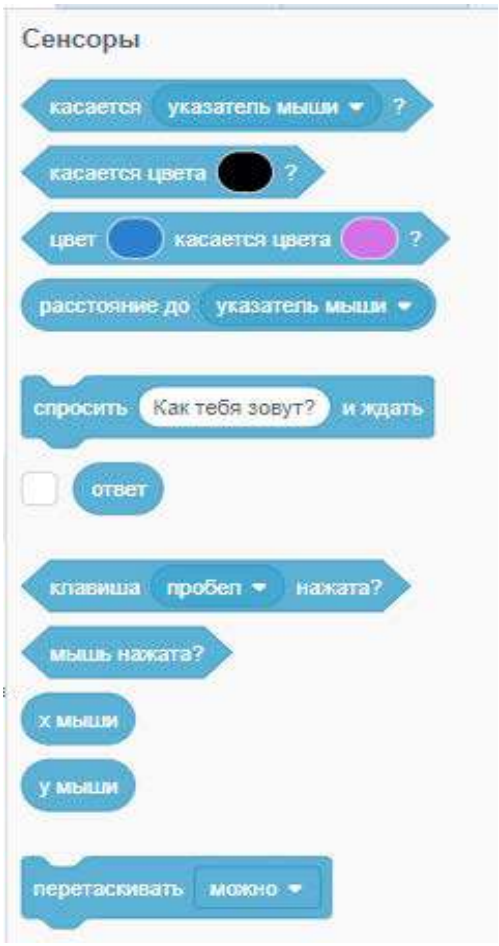

Рис. 154. Вид фрагмента раздела «Сенсоры»

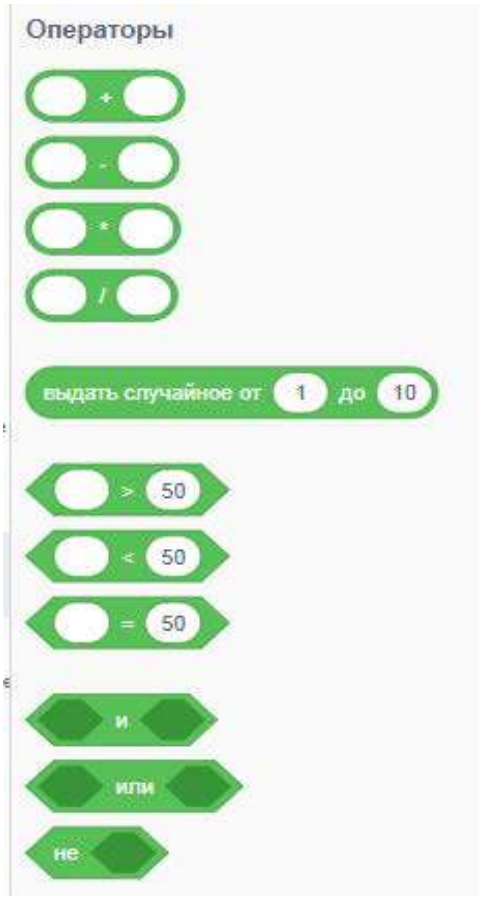

Рис. 155. Вид фрагмента раздела «Операторы»

Раздел **Операторы** содержит команды, которые используются совместно с сенсорами, блоками условий и с переменными. Самый нижний зелёный блок содержит в себе множество математических функций, таких как равно, больше, меньше, алгебраические функции и т. д. В составе данного раздела также множество блоков функций (рис. 155).

Самый последний блок **Другие блоки** позволяет реализовывать подпрограммы.

Методы регистрации данных. Программирование расчётов

Планируемые результаты освоения учебного предмета с описанием универсальных учебных действий, достигаемых обучающимися

Личностные:
- формирование умения самостоятельной деятельности;
- формирование умения работать в команде;
- формирование коммуникативных навыков;
- формирование навыков анализа и самоанализа;
- формирование целеустремлённости и усидчивости в процессе творческой, исследовательской работы и учебной деятельности.

Предметные:
- формирование понятий «алгоритм», «программа»;
- формирование понятий об основных конструкциях языка программирования Python: оператор ветвления if, операторы цикла while, for, вспомогательных алгоритмов;
- формирование основных методов обработки числовой информации с использованием языка программирования;
- формирование основных методов реализации математических расчётов с использованием языка программирования;
- формирование алгоритмического и логического стилей мышления.

Метапредметные:
- формирование умения ориентировки в системе знаний;
- формирование умения выбора наиболее эффективных способов решения задач на компьютере в зависимости от конкретных условий;
- формирование приёмов проектной деятельности, включая умения видеть проблему, формулировать тему и цель проекта, составлять план своей деятельности, осуществлять действия по реализации плана, результат своей деятельности соотносить с целью, классифицировать, наблюдать, проводить эксперименты, делать выводы и заключения, доказывать, защищать свои идеи, оценивать результаты своей работы;
- формирование умения распределения времени;
- формирование умений успешной самопрезентации.

Формы контроля

Во время проведения курса предполагается текущий, промежуточный и итоговый контроль.

Текущий контроль осуществляется регулярно во время проведения каждого лабораторного занятия, заключается в ответе учащихся на контрольные вопросы, демонстрации полученных скриптов в среде Scratch, фронтальных опросов учителем.

Также в тематическом планировании предполагается контрольная работа.

Контрольная работа по проверке полученных навыков по темам «Табулирование функций», «Работа с матрицами»

Протабулируйте следующие функции:
$f(x) = x^3 - 2x + 5$;
$f(x) = 1 - x^5$.
Заданы две матрицы:

$$X = \begin{pmatrix} 2 & 14 \\ 6 & -1 \end{pmatrix}, \quad Y = \begin{pmatrix} -9 & 6 \\ 5 & 7 \end{pmatrix}$$

Найти сумму, произведение матриц X, Y, а также матрицу $6X$.

Содержание и форму организации учебных занятий

Планы учебных занятий

1. Табулирование функций, решение уравнений

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение навыков по созданию программ в среде программирования Python, направленных на изучение табулирования, функции, решения квадратных уравнений.

Метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные).

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Оборудование и материалы: компьютер, презентационное оборудование.

Распределение лабораторных работ: изучение теоретического материала лабораторной работы, выполнение лабораторной работы 1.

2. Работа с матрицами

Планируемые результаты

Предметные: получение навыков по созданию программ в среде программирования Python, направленных на изучение и обработку матриц.

Метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные).

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты: изучение теоретического материала лабораторной работы, выполнение лабораторной работы 2.

3. Физические задачи

Планируемые результаты

Предметные: получение навыков по созданию программ в среде программирования Python, направленных на решение физических задач.

Метапредметные: умение контролировать и корректировать учебную деятельность, способность ставить и формулировать для себя цели действий, прогнозировать результаты, анализировать их (причём как положительные, так и отрицательные).

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению, сформированность навыков сотрудничества со сверстниками; готовность и способность к образованию, в том числе самообразованию.

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты: изучение теоретического материала лабораторной работы, выполнение лабораторной работы 3.

Описание лабораторных работ

Лабораторная работа 1. Табулирование функций, решение уравнений

Теоретическая часть

Для реализации математических расчётов в языке Python можно обращаться к модулю math, который содержит достаточное число математических функций и констант. Данный модуль импортируется командой import:

import math

Полный список математических функций можно вывести командой:

import math

Подробное описание данного модуля можно встретить в описании лабораторных работ к языку Python.

Табулирование функции представляет собой вычисление значений функций при определённом изменении аргумента от некоторого начального значения до некоторого конечного значения с определённым шагом. Итогом будет являться таблица значений. Вопросы, связанные с табуированием функции, возникают при решении достаточно широкого круга задач, например, когда речь идёт о численном решении различных уравнений, поиске минимального и максимального значения функций и т. д. Особенно важно табулирование, если функция не имеет аналитического представления. Также табулирование используется при построении графиков функции.

Предлагается рассмотреть далее пример табулирования функции на языке Python.

При табулировании функции следует придерживаться следующего алгоритма:

1. Определить начальное и конечное значения аргумента, количества точек.
2. Определить шаг — величину, на которую будет изменяться аргумент.
3. Аргумент принять равным начальному значению.
4. Расчёт функции.
5. Увеличить аргумент на значение шага.
6. Повторять действия 4—5 до тех пор, пока не будет подсчитано требуемое количество точек.

Пример 1. Рассмотрим табулирование функции $f(x) = \sin(x) \cdot x$.

В программе выводится два столбца: значение переменной x и значение функции. Используется «\t» для знака табуляции. Благодаря ему значения выстраиваются в два столбца.

```
import math
x = 1.0
while x < 10.0:
   print (x, “\t”, math.sin(x)*x)
   x + = 1.0
```

Результат работы представлен на рисунке 156.

```
=============== RESTART: D:/My_docs/мето.
1.0       0.8414709848078965
2.0       1.8185948536513634
3.0       0.4233600241796016
4.0       -3.027209981231713
5.0       -4.794621373315692
6.0       -1.6764929891935552
7.0       4.598906191031523
8.0       7.914865972987054
9.0       3.7090663671758093
>>>
```

Рис. 156. Результат работы программы

Также в данной лабораторной работе рассматривается вопрос решения квадратного уравнения.

Квадратное уравнение имеет вид: $ax^2 + bx + c = 0$, где $a \neq 0$. Для решения квадратного уравнения можно воспользоваться формулой вычисления корней квадратного уравнения с помощью дискриминанта.

$$D = b^2 - 4ac$$

Далее рассматриваются три случая: $D > 0$

$$x_{1,\,2} = \frac{-b \mp \sqrt{D}}{2a}$$

$$D = 0$$

$$x = \frac{-b}{2a}$$

$D < 0$, то нет корней.

Данные формулы предлагается реализовать в программе на языке Python.

Пример 2. Решить квадратное уравнение $2x^2 + 10x - 3 = 0$.

Программа на языке Python может выглядеть следующим образом:

```
import math
a=2
b=10
c=-3
d=b**2-4*a*c
if d>0:
   x1=(-b+math.sqrt(d))/2/a
   x2=(-b-math.sqrt(d))/2/a
   print("x1=",x1,"\t", "x2=",x2)
if d==0:
   x=(-b)/2/a
   print("x=",x)
if d<0:
   print(«нет корней»)
```

Результат работы программы представлен ниже.

```
=============== RESTART: D:/My_docs/методички/точкироста/2
x1= 0.28388218141501076          x2= -5.283882181415011
>>>
```

Рис. 157. Результат работы программы

Практическая часть

Цель работы: рассмотреть табулирование функции и решение квадратного уравнения на языке программирования Python.

Ход лабораторной работы

Откройте среду разработки Python.

Протабулируйте следующие функции:

1) $f(x) = x^2 + 1$;
2) $f(x) = 2x - x^3$;
3) $f(x) = 2x^4 - x^5$.

Решите следующие квадратные уравнения:

1) $2x^2 + 3x - 16 = 0$;
2) $-5x^2 + 21x + 2 = 0$;
3) $x^2 + 15x + 61 = 0$.

Выводы: в ходе выполнения лабораторной работы вы получили представление о табулировании функции и решении квадратных уравнений на языке Python.

Контрольные вопросы:

1. Какие основные операторы языка Python вы использовали при выполнении лабораторной работы?
2. Для чего выполняется табулировании функции?
3. Какие способы решения квадратных уравнений вы знаете?

Лабораторная работа 2. Работа с матрицами

Теоретическая часть

Для работы с матрицами в языке Python можно использовать следующие варианты:

— использовать списки;
— использовать методы numpy.

Использование списков. В этом случае используются вложенные списки. Каждый элемент списка-матрицы содержит вложенный список. В результате получается структура из вложенных списков, количество которых определяет количество строк матрицы, а число элементов внутри каждого вложенного списка указывает на количество столбцов в исходной матрице.

Например, создадим матрицу размером 3X3.

```
matrix = [[-1, 0, 1], [-1, 0, 1], [0, 1, -1]].
```

Для вывода матрицы можно использовать один стандартный оператор:

```
print(matrix)
```

Такой список можно заполнять случайным образом, используя функцию randint. Ниже представлена программа заполнения матрицы случайным образом и вывода матрицы на экран.

Пример 1

```
from random import randint
n, m = 3, 3
a = [[randint(1, 10) for j in range(m)] for i in range(n)]
```

```
for row in a:
   for x in row:
   print ( “{:4d}”.format(x), end = “” )
print ()
```

Результат работы представлен ниже.

```
>>>
=============== RESTART:
   3   3  10
  10   4   2
   6   7   9
>>>
```

Рис. 158. Результат работы программы

Нумерация такой матрицы начинается с нуля, для к элементу матрицы используется индексация:

a[1][2]

Использовать методов numpy. Существует несколько способов создания массивов NumPy.

Например, numpy.array() можно использовать для создания массива, используя списки в качестве входных данных.

```
import numpy
a = numpy.array([[ 10, -2, 30],[ 0, 50, -10]])
print(a)
```

Также можно использовать функцию numpy.arange(), numpy.matrix()

Для организации обработки матричных величин (сложение, умножение и транспонирование) также используются разные методы. Так, для суммирования матриц NumPy можно использовать традиционную функцию +.

Пример 2

```
import numpy as np
A = np.array([[2, 6], [5, -1]])
B = np.array([[5, -3], [3, 10]])
C = A + B
print(C)
```

Для организации скалярного умножения можно использовать обычную функцию *.

Пример 3

```
import numpy as p
A = p.matrix([[11, 21], [38, 42]])
print(A * 10)
```

А вот для матричного умножения используется функция dot(), которая принимает массивы NumPy в качестве значений параметров и выполняет умножение в соответствии с основными правилами умножения матриц.

Пример 4

```
import numpy as np
a = np.array([[3, 8, 7], [5, -1, 0]])
b = np.array([[1, 5], [2, 1], [3, -1]])
c = a.dot(b)
print(c)
```

Также имеется специальная функция transpose() для получения транспонированной матрицы.

Пример 5

```
import numpy as np
a = np.array([[1, 1], [2, 4], [1, -7]])
print(a.transpose())
```

Практическая часть

Цель работы: рассмотреть работу с матрицами на языке программирования Python.

Ход лабораторной работы

1. Откройте среду разработки Python.
2. Даны две матрицы

$$A = \begin{vmatrix} 3 & -9 & 7 \\ 6 & 1 & 8 \\ -6 & 11 & 5 \end{vmatrix}$$

$$B = \begin{vmatrix} 1 & 6 & -8 \\ 3 & 3 & -9 \\ 0 & 12 & -3 \end{vmatrix}$$

Найдите их сумму, произведение, а также транспонируйте обе матрицы.

3. В таблице 17 представлены результаты ежечасного измерения температуры воздуха на метеостанции.

Таблица 17. Результаты измерений

	0:00	1:00	2:00
01.04.2021	13,7	12,4	12,4
02.04.2021	13,8	12,1	12,8
03.04.2021	12,1	13,2	14,3

Выведите на экран:

1) Показание за три дня в 1:00.
2) Среднее значение температур за указанный период.
3) Найти самую низкую и самую высокую температуру за время измерений.

Контрольные вопросы:

1. Какие основные операторы языка Python вы использовали при выполнении лабораторной работы?
2. Приведите примеры матриц.
3. Какие основные операции над матрицами вы можете назвать?

Лабораторная работа 3. Физические задачи

Теоретическая часть

Компьютер успешно применяется для решения физических задач. При решении физической задачи необходимо установить начальные данные, определяющие связь между параметрами системы. Таким образом, решение состоит из:

1) описания физической задачи и формулировки вопроса, на который нужно получить ответ;
2) выявления основных законов, управляющих процессами в изучаемых системах;
3) получения необходимых данных;
4) формулировки математической задачи;
5) решения математической задачи;
6) анализа результата решения, получения ответа на поставленный вопрос.

Рассмотрим несколько примеров моделирования физических задач с использованием языка Python.

Пример 1. Моделирование физического процесса «движение тела с постоянной скоростью» (равномерное движение тела). Данный процесс характеризуется движением вдоль прямой линии, которую можно совместить с осью *Ox*. Каждую секунду координата *x* тела будет получать одно и то же приращение, поэтому в любой момент времени может быть найдена как:

$x = v_x t$ (1), где v_x — проекция вектора скорости на ось *Ox*.

Также можно рассмотреть случай, когда в начальный момент времени $t_0 = 0$ положение тела не совпадало с началом отсчёта, то уравнение (1) примет вид:

$x(t) = x_0 + v_x t$ (2)

Для реализации данного уравнения также необходимо установить временной промежуток, т. е. задать значения t_0, $t_{кон}$, Δt, v_x, x_0.

Зададим следующие начальные значения:

$t_0 = 0$, $t_{кон} = 20$, $\Delta t = 1$, $x_0 = 0$, $v_x = 5$.

В результате программа на языке Python примет вид:

```
t0=0
tf=20
dt=1
t=t0
x0=0
vx=5
while t<=tf:
    xt=x0+vx*t
    print("{:2d}".format(t),"{:4d}".format(xt), end = "" )
    print()
    t+=dt
```

Результат работы программы выглядит следующим образом.

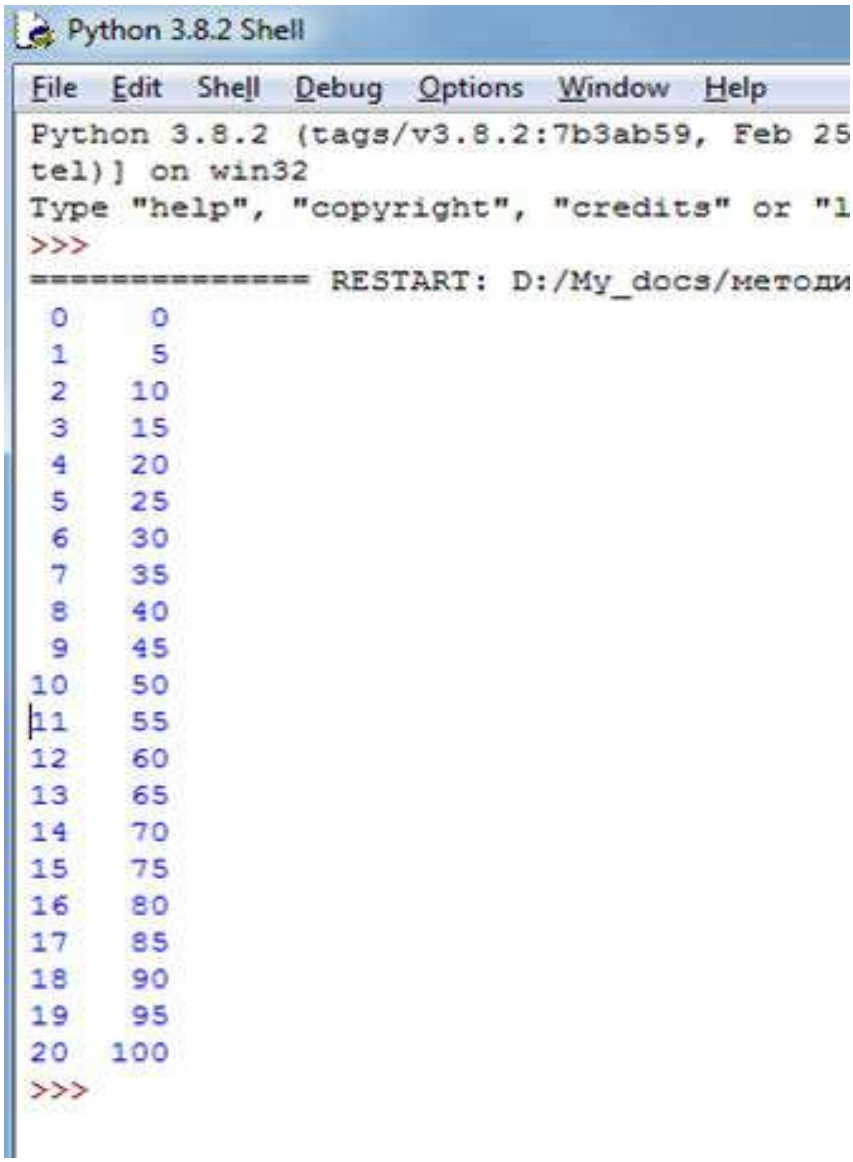

Рис. 159. Результат работы программы

Пример 2. Моделирование физического процесса «движение тела с постоянным ускорением» (равноускоренное движение тела). Движение происходит вдоль прямой, поэтому для его описания рассмотрим изменение одной координаты. Пусть тело движется вдоль оси Oy. Согласно определению ускорения

$$a = \frac{v - v_0}{t} \quad (1).$$

Из (1) выразим v: $v = v_0 + at$ (2).

Так как по условию задачи движение равноускоренное, то при изменении времени скорость получает одно и то же приращение. Перепишем (2) в проекции на выбранное направление оси Oy:

$$v_y = v_{0y} + a_y t \quad (3).$$

В результате уравнение изменения координаты y с течением времени выглядит следующим образом:

$$y(t) = y_0 + v_{0y}t + \frac{a_y t^2}{2} \quad (4).$$

Для реализации данного уравнения также необходимо установить временной промежуток, т. е. задать значения t_0, $t_{кон}$, Δt, v_{0y}, $a_{0y,}$ y_0.

Зададим следующие начальные значения:

$t_0=0$, $t_{кон}=20$, $\Delta t=1$, $y_0=0$, $v_y=5$, $a_y=2$.

В результате программа на языке Python примет вид:

```
t0=0
tf=20
dt=1
t=t0
y0=0
vy=5
ay=2
while t<=tf:
   yt=y0+vy*t+ay*(t**2)/2
   print("{:2d}".format(t),"{:4f}".format(yt), end  =  "" )
   print()
   t+=dt
```

Результат представлен ниже:

```
>>>
=============== RESTART: D:/My_
 0 0.000000
 1 6.000000
 2 14.000000
 3 24.000000
 4 36.000000
 5 50.000000
 6 66.000000
 7 84.000000
 8 104.000000
 9 126.000000
10 150.000000
11 176.000000
12 204.000000
13 234.000000
14 266.000000
15 300.000000
16 336.000000
17 374.000000
18 414.000000
19 456.000000
20 500.000000
>>>
```

Рис. 160. Результат работы программы

Практическая часть

Цель работы: рассмотреть работу по реализации физических задач на языке программирования Python.

Ход лабораторной работы

1. Откройте среду разработки Python.
2. Рассмотрите равномерное движение дела, установив следующие значения:

1) $t_0 = 0, t_{кон} = 50, \Delta t = 0{,}1, x_0 = 0, v_x = 3.$

2) $t_0 = 1, t_{кон} = 30, \Delta t = 0{,}1, x_0 = 2, v_x = 10.$

Рассмотрите равноускоренное движение дела, установив следующие значения:

$t_0 = 0, t_{кон} = 20, \Delta t = 0{,}1, y_0 = 3, v_y = 5, a_y = 2.$

$t_0 = 0, t_{кон} = 50, \Delta t = 0{,}1, y_0 = -1, v_y = 2, a_y = 6.$

Контрольные вопросы:

1. Какие основные операторы языка Python вы использовали при выполнении лабораторной работы?
2. Какие физические задачи вы рассмотрели при выполнении лабораторной работы?

Дидактические материалы

Матрицы

Определение. Матрицей размера *m* X *n* называется таблица, образованная из элементов некоторого множества (например, чисел или функций) и имеющая *m* строк и *n* столбцов.

Если $m \neq n$, то матрицу называют **прямоугольной**. Если $m = n$, то матрицу называют **квадратной**, порядка *n*. Элементы, из которых составлена матрица, называются **элементами** матрицы.

Вид матрицы 3 X 3 представлен ниже:

$$A = \begin{pmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{pmatrix}$$

Матрицу можно обозначить: $A = (a_{ij}), (i = \overline{1, m}, j = \overline{1, n})$.

Определение. Матрицу называют нулевой, если все её элементы равны нулю.

Определение. Две матрицы *A* и *B* считаются равными, если они одинакового размера, и элементы, стоящие в *A* и *B* на одинаковых местах, равны между собой, т. е. $a_{ij} = b_{ij}$.

Определение. Элементы $a_{11}, a_{22}, \ldots, a_{kk}$ (где $k = \min\{m, n\}$) будем называть элементами главной диагонали матрицы.

Определение. Диагональная матрица, у которой все элементы главной диагонали равны 1, называется единичной. Единичная матрица 2×2 представлена ниже:

$$A = \begin{pmatrix} 1 & 0 \\ 0 & 1 \end{pmatrix}.$$

Опишем основные операции над матрицами:

1. Умножение матрицы на число.

Произведением матрицы $A = (a_{ij})$ на число k называется такая матрица $B = (b_{ij})$, элементы которой произведениям соответствующих матрицы A на число k, т. е. $b_{ij} = k \cdot a_{ij}$.

Обозначают: kA

2. Сумма матриц.

Суммой двух матриц $A = (a_{ij})$ и $B = (b_{ij})$ одинакового размера, называется такая матрица $C = (c_{ij})$, элементы которой равны суммам соответствующих элементов матриц A и B, т. е. $c_{ij} = a_{ij} + b_{ij}$.

Обозначают: $A+B$.

3. Умножение матриц

Пусть $A = (a_{ij})$ — матрица размера $m \times n$, $B = (b_{ij})$ — матрица размера $n \times k$ (т. е. количество столбцов в матрице A совпадает с количеством строк матрицы B). Произведением матрицы A на матрицу B называется матрица $C = (c_{ij})$ размера $m \times k$ такая, что каждый её элемент c_{ij} является произведением i-й строки матрицы A на j-й столбец матрицы B, т. е.

$$c_{ij} = a_{i1} \cdot b_{1j} + a_{i2} \cdot b_{2j} + a_{i3} \cdot b_{3j} + \ldots + a_{in} \cdot b_{nj}.$$

Обозначают: $A \cdot B$, AB.

4. Транспонирование матрицы

Пусть A — матрица размера $m \times n$. Матрица размера $n \times m$, полученная из A заменой каждой её строки столбцом с тем же номером, называется транспонированной к A и обозначается A^T. Операция нахождения матрицы A^T называется транспонированием матрицы A.

Вопросы искусственного интеллекта

Планируемые результаты освоения учебного предмета с описанием универсальных учебных действий, достигаемых обучающимися

Личностные:
- формирование умения самостоятельной деятельности;
- формирование умения работать в команде;
- формирование коммуникативных навыков;
- формирование навыков анализа и самоанализа;
- формирование целеустремлённости и усидчивости в процессе творческой, исследовательской работы и учебной деятельности.

Предметные:
- формирование основных понятий математической логики;
- формирование понятий об основных конструкциях языка Prolog;
- формирование знаний об основных предикатах языка Prolog;
- формирование знаний об основных типах и структурах данных.

Метапредметные:
- формирование умения ориентировки в системе знаний;
- формирование умения выбора наиболее эффективных способов решения задач на компьютере в зависимости от конкретных условий;
- формирование приёмов проектной деятельности, включая умения видеть проблему, формулировать тему и цель проекта, составлять план своей деятельности, осуществлять действия по реализации плана, результат своей деятельности соотносить с целью, классифицировать, наблюдать, проводить эксперименты, делать выводы и заключения, доказывать, защищать свои идеи, оценивать результаты своей работы;
- формирование умения распределения времени;
- формирование умений успешной самопрезентации.

Формы контроля

Во время проведения курса предполагается текущий, промежуточный и итоговый контроль.

Текущий контроль осуществляется регулярно во время проведения каждого лабораторного занятия, заключается в ответе учащихся на контрольные вопросы, демонстрации полученных скриптов в среде Scratch, фронтальных опросов учителем.

Также в тематическом планировании предполагается контрольная работа.

Контрольная работа по проверке полученных навыков по темам «Встроенные предикаты языка программирования Prolog», «Понятие рекурсивного алгоритма, виды рекурсии».

1. Вычислить:

1) $-6x + \frac{1}{3}y + 13{,}7;$

2) $(2z + 4) / 4y.$

2. Даны три измерения прямоугольного параллелепипеда. Найти площадь боковой поверхности и объём.

3. Для интерактивно вводимого натурального *N* вывести на экран в строчку через пробел числа: *N, N−1, ...,2,1, 2, 3, ..., N−1, N*.

4. Вывести на экран максимальную цифру в числе.

Содержание и форма организации учебных занятий

Планы учебных занятий

1. Основные понятия языка программирования Prolog: предикаты, операции над предикатами

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение навыков описания предикатов, операций над предикатами в языке Prolog, разработки правил и первых программ на языке Prolog.

Метапредметные: умение проанализировать поставленную задачу и те условия, в которых она должна быть реализована, рассматривать разные точки зрения и выбрать правильный путь реализации поставленных задач.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; ответственное отношение к учению, готовность и способность обучающихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию.

2. Встроенные предикаты языка программирования Prolog

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение навыков работы со встроенными предикатами при написании программ на языке Prolog.

Метапредметные: умение проанализировать поставленную задачу и те условия, в которых она должна быть реализована, рассматривать разные точки зрения и выбирать правильный путь реализации поставленных задач; выстраивать логические рассуждения, делать умозаключения и собственные выводы.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; ответственное отношение к учению, готовность и способность обучающихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию; формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками, детьми старшего и младшего возраста, взрослыми в процессе образовательной, общественно полезной, учебно-исследовательской, творческой и других видов деятельности.

3. Понятие рекурсивного алгоритма, виды рекурсии. Реализация рекурсивных алгоритмов в языке программирования Prolog

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение навыков работы с рекурсивными правилами в языке Prolog, написания программ, реализующих рекурсивные алгоритмы.

Метапредметные: умение проанализировать поставленную задачу и те условия, в которых она должна быть реализована, рассматривать разные точки зрения и выбирать правильный путь реализации поставленных задач; выстраивать логические рассуждения, делать умозаключения и собственные выводы.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; ответственное отношение к учению, готовность и способность обуча-

ющихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию; формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками, детьми старшего и младшего возраста, взрослыми в процессе образовательной, общественно полезной, учебно-исследовательской, творческой и других видов деятельности.

Описание лабораторных работ

Лабораторная работа 1. Основные понятия языка программирования Prolog: предикаты, операции над предикатами

Теоретическая часть

Определение. Пусть n — неотрицательное целое число, тогда $p: M_1 \times M_2 \times \ldots \times M_n \to \{0,1\}$ называется ***n*-местным предикатом**, определённым на множестве $D(p) = M_1 \times M_2 \times \ldots \times M_n$.

P называется именем предиката, множество $D(p)$ — **областью истинности предиката**, множество $D^+(p) = \{(m_1, m_2, \ldots m_n) \in D(p) : p(m_1, m_2, \ldots m_n) = 1\}$ называется **множеством истинности** предиката p. Число n называется **рангом** предиката p и обозначается $R(p)$.

Каждое из множеств M_i, входящее в область определения предикатов, может быть одним из следующих пяти видов.

integer	$[-32768, 32767] \in Z$
real	$[10^{-307}; 10^{308}] \in R$
char	Множество символов ASCII, заключённых в апострофы
string	Множество строк, заключенных в кавычки
symbol	Множество, состоящее из строк и идентификаторов

Утверждение $(a_1, a_2, \ldots, a_n) \in D^+(p)$ называется **фактом** и записывается в виде $p(a_1, a_2, \ldots, a_n)$. **Запись факта завершается обязательно точкой!**

Структура программы на языке Prolog

Domains
[описание типов]
Predicates
[описание имён предикатов]
Clauses
[описание множеств истинности предикатов]
Goal/ goal
[цель согласования]

Переменные в ТР аналогичны неопределённым местоимениям «некто», «нечто» в естественном языке. Запись переменной обязательно начинается с большой латинской буквы. Переменные могут быть трёх видов: свободные, связанные и анонимные. **Свободная** переменная не имеет значения, а **связанная** переменная имеет некоторое значение. **Анонимная** переменная используется для обозначения места аргумента в предикате, если его значение не важно.

Операции над предикатами

В ТР в качестве операций над предикатами используется логические операции: отрицание, конъюнкция и дизъюнкция.

Пусть имеются два предиката: p, $R(p) = n$, $D(p)$ — его область определения, $D^+(p)$ — его область истинности, q, $R(q) = m$, $D(q)$ — его область определения, $D^+(q)$ — его область истинности,

Определение. **Отрицанием** предиката $p(X_1, X_2,..., X_n)$ является предикат 1 — $p(X_1, X_2,..., X_n)$, который обозначается $not(p(X_1, X_2,..., X_n))$.

$R(not(p))=R(p)$

$D(not(p))=D(p)$

$D^+(not(p))=D(p)\setminus D^+(p)$

Определение. **Конъюнкцией** двух предикатов p и q является их произведение $p \times q$ и обозначается p, q.

$R(p,q)=m+n-k$, где k — количество общих переменных в предикатах p и q.

$D(p,q)=D(p) \cup D(q)$.

$D^+(p,q)=D(p) \cap D(q)$.

В разделе clauses программист задаёт множества истинности предикатов из раздела predicates, на основе которых Пролог согласует цель. Совокупность этих множеств называется **базой данных** или **базой знаний**. Первый способ задания базы знаний состоит в перечислении фактов, второй — использовании правил.

Пусть все переменные m-местного предиката $q(X_1, X_2,...,X_m)$ являются общими с $m + n$-местным предикатом $p(X_1, X_2,...,X_m, Y_1, Y_2, ...Y_n)$. **Правило** есть указание Прологу найти упорядоченный набор $(a_1, a_2,...,b_1, b_2, ...,b_{n\ я}) \in D^+(p)$ и считать набор $(a_1, a_2,...,a_n)$ элементом множества $D^+(q)$.

Обозначается $q(X_1, X_2,...,X_m)$:-$p(X_1, X_2,...,X_m, Y_1, Y_2,...,Y_n)$. Предикат p называется телом правила, предикат q — заголовком. **Запись правила также завершается точкой.**

Практическая часть

Цель работы: рассмотреть основные понятия языка Prolog, ввести понятие предикат, операции над предикатами: отрицание, конъюнкция, дизъюнкция.

Ход лабораторной работы

1. Описать в виде фактов следующие высказывания.

1) Петров — ученик 8 класса. Иванов — ученик 9 класса. Сидоров — ученик 7 класса.

2) Полина любит яблоки. Марина любит персики. Жанна любит апельсины.

3) Москва — столица России. Лондон — столица Англии. Париж — столица Франции.

2. В программе заданы следующие факты

мужчина(X)

женщина(X)

родитель(X,Y).

Написать правила для следующих отношений

является_матерью(X)

является_отцом(X)

является_сыном(X)

общие_родители(X)

3. Описать в виде правила следующие родственные отношения: сестра, деверь (брат мужа), тёща (мать жены), племянница (дочь брата или сестры).

Выводы: в ходе выполнения лабораторной работы вы получили представление об основных понятиях языка Prolog.

Контрольные вопросы:

1. Дайте определение предиката.
2. Приведите примеры предикатов, фактов.
3. Какие операции над предикатами в языке Prolog вы можете назвать?

Лабораторная работа 2.
Встроенные предикаты языка программирования Prolog

Теоретическая часть

В Prolog есть особый класс предикатов, которые не надо описывать в разделе predicates, а их множества истинности не надо задавать в разделе clauses, так как Турбо Пролог «знает» о них все. Такие предикаты называются **встроенные**.

Встроенные предикаты можно разделить на основные группы:

1. Встроенные предикаты для ввода данных

readint (переменная)	Ввод целого числа
readreal (переменная)	Ввод действительного числа
readchar (переменная)	Ввод символа
readln (переменная)	Ввод строки

2. Встроенные предикаты для вывода данных

write(список связанных переменных)	Вывод на экран значений переменных из списка

3. Встроенные предикаты управления экраном

Nl	Перевод курсора на новую строку
clearwindow	Очистка экрана

4. Встроенные математические предикаты

Sin(X)	Синус X
Cos(X)	Косинус X
Tan(X)	Тангенс X
Arctan(X)	Арктангенс X
Abs(X)	Модуль X
Exp(X)	Экспонента X
Ln(X)	Натуральный логарифм X
Log(X)	Десятичный логарифм X

5. Встроенные арифметические предикаты
+, -, *, /, div, mod
6. Бинарные отношения
>, <, =, >=, <=, =

Практическая часть

Цель работы: рассмотреть возможности ввода-вывода данных с помощью встроенных предикатов языка Prolog, построение математических выражений, вычислительных программ.

Ход лабораторной работы

1. Запишите правила, являющиеся решениями следующих заданий:

а) Даны два числа a и b, получить их сумму, разность, произведение.

б) Дана длина ребра куба. Найти его объём и площадь боковой поверхности.

в) Даны радиус основания и высота цилиндра. Найти его объём и площадь поверхности.

г) Даны стороны a, b параллелограмма и угол между ними. Найти площадь параллелограмма.

2. Вычислить:

1) $2x + \frac{1}{3}y + 5.6$;

2) $(2x + 8y + 4) / 2$;

3) $y - x^2$;

4) $x^2 + xy + y^2$;

5) $\sin(x) + \cos(3y)$;

6) $5x + \sqrt{34z} + \frac{y}{3}$.

3. Написать программу решения квадратного уравнения.

Выводы: в ходе выполнения лабораторной работы вы получили представление об основных встроенных предикатах языка Prolog.

Контрольные вопросы:

1. Какой предикат называется встроенным в языке Prolog?
2. Какие встроенные предикаты языка Prolog вы можете назвать?
3. Для чего используется предикат readln?

Лабораторная работа 3.
Понятие рекурсивного алгоритма, виды рекурсии. Реализация рекурсивных алгоритмов в языке программирования Prolog

Теоретическая часть

Правило называется **рекурсивным**, если в его теле находится его же заголовок. В Прологе существует два вида рекурсивных правил: **простое** и **обобщённое**.

Рекурсивное правило называется **простым**, если его заголовок — нульместный предикат. Правило называется **обобщённым**, если его заголовок — не нульместный предикат.

Обобщенные рекурсивные правила бывают двух видов:

- **Рекурсивное правило с непосредственным согласованием** — если его не нульместный заголовок является последним в теле правила.
- **Рекурсивное правило с отложенным согласованием** — если его не нульместный заголовок не является последним в его теле.

При согласовании обобщённых рекурсивных правил Пролог использует стек.

Пример 1. Вычислить *n*!
Вспомним рекурсивное определение факториала:
1) $N = 1$, $n! = 1$.
2) $N > 1$, $n! = n*(n - 1)!$

Программа имеет вид:

```
Predicates
fact(integer,integer)
do
clauses
fact(1,1):-!.
fact(X,Y):-X1=X-1,fact(X1,Y1),Y=Y1*X.
do:-write("N= "), readint(N), fact(N,FN), write("N!= ", FN).
goal do.
```

Практическая часть

Цель работы: приёмы построения рекурсивных программ, применение различных видов рекурсий для решения задач на языке Prolog.

Ход лабораторной работы

1. Для интерактивно вводимого натурального *N* вывести на экран в строчку через пробел числа: 1, 2, 3, ..., *N*.
2. Для интерактивно вводимого натурального *N* вывести на экран в строчку через пробел числа: *N*, *N* − 1, ...,2,1.
3. Для интерактивно вводимого натурального *N* вывести на экран в строчку через пробел числа: 1, 2, 3, ..., *N* , *N* − 1, ...,2,1.
4. Вычислить сумму квадратов первых *N* чисел, где *N* вводится с клавиатуры.
5. Вывести на экран НОД двух целых чисел.
6. Вывести на экран все числа от 1 до 2000, делящиеся на 8 и заканчивающиеся 6.

Выводы: в ходе выполнения лабораторной работы вы получили представление о реализации рекурсивных правил в языке Prolog.

Контрольные вопросы:

1. Дайте определение рекурсии в языке Prolog.
2. Приведите примеры рекурсивных процессов.
3. Какие виды рекурсий в языке Prolog вы можете назвать?

Дидактические материалы

Вопросы искусственного интеллекта

В настоящее время на практике применяются различные подходы к обработке информации, формируются и развиваются новые технологии. Одной из них является технология искусственного интеллекта. Термин искусственный интеллект (ИИ) многозначен, он впервые был введён Джоном Маккарти в 1956 г. На английском языке ИИ означает — Artificial Intelligence (AI). Необходимо отметить, что термин ИИ не связан с человеческим сознанием и не имеет «антропоморфной окраски». Этимология слова «интеллект» (лат. Intellectus) означает в переводе с латинского «понимание», «рассудок» и происходит из сочетания слов intelligere, означающего «воспринимать», «познавать», «мыслить», и legere — «собирать», «читать».

Искусственный интеллект представляет собой «свойство интеллектуальных систем выполнять творческие функции, являющиеся прерогативой человека, наука и технология создания интеллектуальных машин, особенно интеллектуальных компьютерных программ». ИИ не имеет взаимосвязи с биологически оправданными методами. Современные системы ИИ ограничены областями определения, например, программа игры в шахматы не может отвечать на вопросы, посвящённые литературному творчеству. Вместе с тем установлено, что структура интеллектуальной системы состоит из трёх блоков — базы знаний, процессора и интеллектуального интерфейса вычислительной машины, не использующего специальных программ ввода-вывода данных.

В современной философии не решён вопрос о природе и статусе человеческого интеллекта, поэтому не существует и точного критерия достижения компьютерами «разумности», хотя на заре искусственного интеллекта был предложен ряд гипотез, например, тест Тьюринга, общепринятая его интерпретация может быть сформулирована следующим образом: «Человек взаимодействует с одним компьютером и одним человеком. На основании ответов на вопросы он должен определить, с кем он разговаривает: с человеком или компьютерной программой. Задача компьютерной программы — ввести человека в заблуждение, заставив сделать неверный выбор».

Ещё один подход основан на гипотезе Ньюэлла—Саймона, сформулированной в 1976 г. Алленом Ньюэллом и Гербертом Саймоном. Гипотеза утверждает, что «физическая символьная система имеет необходимые и достаточные средства для произведения основных интеллектуальных операций». Иначе говоря, без выполнения символьных вычислений невозможны осмысленные действия, способность выполнять символьные вычисления достаточна для того, чтобы выполнять осмысленные действия. Предполагается, что если какой-то объект, человек или машина действуют осмысленно, то они выполняют символьные вычисления. Наоборот, компьютер способен к подобным вычислениям, следовательно, на его основе может быть создан искусственный интеллект. Данная гипотеза подвергается критике, однако, несмотря на это, исследования в области ИИ сконцентрированы в области обработки символов. Вместе с тем, вне зависимости от истинности данной гипотезы, символьные вычисления — основа программирования компьютеров и их значение очевидно.

Поэтому, несмотря на наличие множества концепций понимания задач ИИ и создания интеллектуальных информационных систем, можно выделить два основных подхода к разработке систем ИИ:

- нисходящий ИИ (англ. Top-Down AI), семиотический — создание экспертных систем, баз знаний и систем логического вывода, имитирующих высокоуровневые психические процессы: мышление, рассуждение, речь, эмоции, творчество и т. д.;
- восходящий ИИ (англ. Bottom-Up AI), биологический — изучение нейронных сетей и эволюционных вычислений, моделирующих интеллектуальное поведение на основе биологических элементов, а также создание соответствующих вычислительных систем, таких как нейрокомпьютер или биокомпьютер.

Последний подход, строго говоря, не относится к науке об ИИ в смысле, определения, данного Джоном Маккарти, — их объединяет только общая конечная цель.

Несколько слов об истории искусственного интеллекта, в том числе в нашей стране. В России коллежский советник С. Н. Корсаков (1787—1853) предложил в 1837 г. применить перфокарты во врачебном деле с целью диагностики заболеваний и формирования схемы лечения. По сути дела, эти карты можно рассматривать как некоторую систему искусственного интеллекта.

В 1964 г. С. Ю. Маслов подготовил статью под названием «Обратный метод установления выводимости в классическом исчислении предикатов». В ней впервые предлагался метод автоматического поиска доказательства теорем в исчислении предикатов.

В 1966 г. В. Ф. Турчин разработал язык РЕФАЛ. РЕФАЛ — это кириллическая аббревиатура английской фразы REcursive FUnction Algorithmic Language, означающей в переводе на русский язык — Язык программирования рекурсивных функций.

В 1972 г. профессор университета в Марселе Ален Колмероэ создал язык Пролог, основанный на исчислении предикатов. Аббревиатура Пролог означает ПРОграммирование в ЛОГике. Необходимо отметить, что этот язык определил существенную эпоху в методах обработки информации. Появление этого языка позволило реализовать на практике нисходящий подход в теории искусственного интеллекта. В литературе известно много применений данного языка в различных сферах деятельности человека, в том числе и в сфере образования.

Языки программирования РЕФАЛ и ПРОЛОГ отражают так называемую декларативную парадигму программирования. Программа, написанная на этих языках, отражает описание — декларацию задачи. Иной подход или парадигма связаны с описанием алгоритма решения задачи, они характерны для языков С, С++, Python, JAVA, PASCAL, SCRATCH, относящихся к императивной парадигме. В школьном курсе информатики изучают в основном императивный, алгоритмический подход.

Для подробного рассмотрения основ логического программирования необходимо обратиться к основам математической логики и обозначить некоторые понятия и термины.

Константы, в этом качестве могут быть любые элементы любого множества. Например, числовые и литеральные константы: 2; 35; 3,14156 — числовые константы; *a*, n, PROLOG — литеральные константы.

Переменные — это символы, которые могут принимать любые значения из множества констант.

Функции — представляют собой отображение одного множества в другое. Примером может быть функция ученик (*x*) = *y*, ставящая в соответствие множество учащихся и множество классов школы.

Константы, переменные и функции в математической логике объединены общим названием — терм.

Над объектами могут быть определены отношения. Отношение определяет совокупность элементов из предметной области и представляет собой отображение из множества элементов в множество, состоящее из двух элементов {ИСТИНА, ЛОЖЬ}. Например, отношение папа (*x, y*) определяет совокупность пар (*x, y*), таких, что элементы множества людей *x* и *y* находятся отношении родства папа. Совокупность пар (*x, y*) есть множество матерей и детей. В математической логике даются имена, называемые предикатными символами, а сами отношения называются предикатами. В примере папа — предикатный символ. Предикат определяется следующим образом.

Предикат — это выражение вида $P(t_1, t_2,...t_m)$, где t_i — терм, а P — m-арный предикатный символ. Предикат $P(t_1, t_2,...t_m)$ читается как «P — справедливо для совокупности $(t_1, t_2,...t_m)$».

В логике определены логические связки: &, □, ¬, □, □. Они соответственно означают «и», «или», «не», «если», «тогда и только тогда». Эти связки можно использовать для образования логических формул. Логические связки могут быть выражены друг через друга.

В математической логике наряду со связками вводится понятие квантора. Существует квантор общности — ∀ и квантор существования — ∃. Кванторы определяют пределы изменения переменных. Логическое выражение, стоящее после квантора, называется областью действия данного квантора.

Приведённые понятия позволяют сформулировать понятие **формула**.

Формула — это предикат или выражение, составленное из формул с помощью логических связок и кванторов.

Необходимо отметить, что определение понятия формулы рекурсивно, в нём присутствует понятие «формула».

С помощью понятия «формула» можно определить **предложение**.

Предложение — это формула, в которой каждая переменная находится в пределах действия квантора общности. В целях лаконичности в записи предложений кванторы общности опускаются.

Примером предложения может служить выражение:

бабушка (*x, y*) ← мама(*x, z*),мама(*z, y*).

Содержательно эта запись означает, что бабушка — это «мама мамы».

Используя введённые понятия формулы и предложения, можно определить любое отношение, выражающее свойства элементов. Это означает, что если для некоторой совокупностью элементов задана формула или предложение, принимающее значение ИСТИНА, то эта совокупность является множеством элементов, задаваемых данной функцией или предложением. Таким образом, может быть сформулирована теория, позволяющая определять множества, она находится в основе исчисления предикатов, которое, в свою очередь, является базой языка логического программирования ПРОЛОГ.

Как известно, множество может быть задано с помощью перечисления элементов множества либо определением свойств его элементов. В первом случае для описания элементов множества может быть использована совокупность фактов, являющихся синтаксической конструкцией языка. Вместе с тем часто бывает необходимо задать определённые связи и отношения между объектами. Такие связи и отношения задаются при помощи правил, которые также присутствуют в числе синтаксических элементов ПРОЛОГа. Правила позволяют сократить число фактов, база знаний (так обычно называют программу на ПРОЛОГе) становится более лаконичной. Правило можно построить, применяя принцип разделения исходной задачи на более простые. Такой подход называется декомпозицией. Процесс декомпозиции завершается, когда отношения связывают зафиксированные в базе знаний объекты. В качестве примера приведём построение родственных отношений:

бабушка (*x, y*) ← мама (*x, z*), мама (*z, y*);
бабушка (*x, y*) ← мама (*x, z*), папа (*z, y*);
дедушка (*x, y*) ← папа (*x, z*), папа (*z, y*);
дедушка (*x, y*) ← папа (*x, z*), мама (*z, y*).

Если ввести правило, определяющее понятие «родитель»:

родитель (*x, y*) ← мама (*x, y*);
родитель (*x, y*) ← папа (*x, y*).

тогда понятия дедушка и бабушка можно определить проще:

бабушка (*x, y*) ← мама (*x, z*), родитель (*z, y*);
дедушка (*x, y*) ← папа (*x, z*), родитель (*z, y*).

Если к данной базе знаний добавить несколько фактов, определяющих мам и пап, то получается база знаний «семья»:

мама (Саша, Федя);
папа (Серёжа, Федя);
мама (Оля, Саша);
папа (Коля, Саша);
мама (Люда, Серёжа);
папа (Федя, Серёжа);
родитель (*x, y*) ← мама (*x, y*);
родитель (*x, y*) ← папа (*x, y*);
бабушка (*x, y*) ← мама (*x, z*), родитель (*z, y*);
дедушка (*x, y*) ← папа (*x, z*), родитель (*z, y*).

Данной базе знаний с помощью средств ПРОЛОГа можно задать вопросы. Например, «Является ли Саша мамой Феди?» или «Является ли Оля бабушкой Феди?».

? мама (Саша, Федя);

Ответ — ИСТИНА;

? бабушка(Оля, Федя);

Ответ — ИСТИНА.

Выполняя программу, система ПРОЛОГ установила, что в рассматриваемой семье Оля — бабушка Феди, а Саша его мама. Компьютер выполнил условия теста Тьюринга, в соответствии с которым он сделал правильный выбор и его можно признать «умным».

Возможными направлениями применения логического программирования является работа с базами данных. Можно достаточно естественным образом описывать запросы, комбинируя предикаты, причём научить писать такие запросы можно даже человека, совершенно незнакомого с логическим программированием. Логическое программирование применяют и для решения проблем структуризации и разбора текста на основе грамматических правил.

Но в задачах создания качественного текста необходимо обратиться к технологии применения нейросетей, которая будет рассмотрена позднее.

Анализ и генерация предложений является сложной задачей. При создании и восприятии текста человек основывается не только на наборе слов и их семантике, но и на собственном видении окружающего мира. Например, если в базе данных содержится факт «Петя отремонтировал Оле компьютер», то на вопрос «Разбирается ли Петя в компьютерах?» нельзя получить ответ от программы, хотя кажется, что решение «очевидно». Из-за низкой скорости анализа и особенностей применения логических языков их не удаётся применять для решения задачи поиска котиков на картинке. Однако задачи синтаксического разбора, анализа текстов и тому подобные на логических языках удается решать.

Ещё одно применение логического программирования — поиск решений. Если есть набор знаний о некоторой системе и нужно понять возможность обойти её защиту, то это можно лаконично описать при помощи логических языков и начать процесс поиска решений. Помимо задач перебора легко решить те, где требуются логические рассуждения, анализ кода или естественного текста.

Язык Пролог можно использовать для создания собственного языка со своими правилами, позволяющими организовать взаимодействие между различными предметными областями.

Важно отметить, что программы на логических языках оказываются неэффективны, но удобны, особенно если речь не идёт об ограниченных, специализированных решениях. Программа на императивном языке исполняется быстрее аналогичной программы на логическом, но затраты на написание программ на Прологе ниже в разы.

В нашей стране в курсе информатики активно применялся Пролог-Д, основанный на принципах математической логики, представлял собой нисходящий подход к технологиям «искусственного интеллекта», он был реализован на школьных компьютерах и был доступен учащимся, изучающим информатику. Особенностью трансляторов этого языка программирования было использование родного языка учащихся, в частности, русского языка, как основы построения интерфейса взаимодействия пользователя и компьютера.

Изучение декларативных языков программирования, таких как Пролог, по оценке ряда методистов-исследователей, целесообразно как второго по порядку языка после изучения языков императивной парадигмы. Это мнение основано на нескольких аргументах. Во-первых, изучение логического подхода предоставляет возможность научиться мыслить иначе, более глубоко освоить процесс разработки. Во-вторых, важно рассмотреть альтернативные методы решения задачи, используя Пролог. В-третьих, бывает удобно создать на логическом языке прототип, реализующий требуемые функции, и затем на его

основе создавать полноценное решение. В-четвёртых, даже если логические языки редко применяются в реальной практике, но декларативный подход встречается достаточно часто, так как многие языки реализуют в себе логические или функциональные элементы.

Восходящая парадигма была сформулирована в 2009 г. на основе идеологии нейронных сетей, или нейросетей. Следует отметить, что она получила в настоящее время достаточно широкое распространение. Нейросети в настоящее время применяются для диагностики заболеваний, в финансовой и банковской сферах, на транспорте для управления беспилотными автомобилями, в промышленности, сельском хозяйстве, образовании и сфере искусств.

Нейронные сети являются одним из векторов развития исследований в области разработки искусственного интеллекта (ИИ). В основе этого подхода находится попытка моделировать нервную систему человека, включая ее способность к самообучению. Это можно в определённой степени рассматривать как попытку смоделировать работу мозга человека.

Искусственные нейронные сети — это вычислительные системы, способные к самообучению, постепенному увеличению своей производительности. Структура нейронной сети включает:

— искусственные нейроны, являющиеся элементарными, связанными друг с другом единицами;
— синапс — представляет собой связь, используемую для передачи-получения информации между нейронами;
— сигнал — это непосредственно передаваемая информация.

Сфера применения искусственных нейронных сетей постоянно расширяется. В настоящее время они применяются в следующих областях:

— машинное обучение (machine learning), являющееся одним из видов искусственного интеллекта. Оно основано на обучении ИИ многочисленными примерами однотипных задач. Машинное обучение активно используют различные поисковые системы, такие как Google, Яндекс, BING, BAIDU. На основе многочисленных поисковых запросов, вводимых пользователями ежедневно в этих поисковых системах, алгоритмы данных систем обучаются определять наиболее релевантные данные, чтобы пользователи находили именно то, что им необходимо;
— в робототехнике нейронные сети применяются для создания различных алгоритмов, обеспечивающих их функционирование, например, распознавание речи и образов, синтез речи и т. д.;
— создатели компьютерных систем применяют искусственные нейронные сети для выполнения параллельных вычислений;
— используя нейронные сети, можно решать сложные математические задачи.

Принята следующая классификация нейронных сетей:

- свёрточные нейронные сети
- реккурентные нейронные сети
- нейронная сеть Хопфилда

Свёрточные сети — самый популярный вид искусственных нейронных сетей. Они эффективно применяются в системах распознавания визуальных объектов и обработке речи. Такие нейронные сети легко масштабируются и используются для распознавания образов, любого разрешения, поскольку в этих сетях применяются трёхмерные нейроны. Внутри одного слоя нейроны связаны лишь небольшим полем, называемым рецептивным слоем. Нейроны других слоёв связаны с помощью пространственной локализации. Функционирование совокупности таких слоёв реализуют специальные нелинейные фильтры, учитывающие всё большее количество пикселей.

Рекуррентными называются нейронные сети, соединения между нейронами которых образуют ориентированный цикл. Они имеют следующие особенности:

- любое соединение имеет свой вес, являющийся вместе с тем приоритетом;
- узлы могут быть либо вводные, либо скрытые;
- в рекуррентной нейронной сети информация передаётся слой за слоем по прямой и непосредственно между нейронами;
- особенность рекуррентной нейронной сети состоит в наличии особой «области внимания», в которой машине определяют фрагменты данных, требующих дополнительной обработки.

Рекуррентные нейронные сети применяются в распознавании и обработке текстовых данных (в частности, на базе таких сетей работают Google-переводчик, голосовой помощник Apple Siri).

Нейронная сеть Хопфилда представляет собой самоассоциативную сеть, способную выполнять функцию памяти. Основное её применение — восстановление образца, сохранённого прежде нейронной сетью, по представляемому на входе искажённому образцу.

Область применения нейронных сетей в настоящее время очень широка.

Например, «умные» плейлисты музыки (например, Яндекс.Музыка или YouTube подбирают уникальный плейлист, исходя из того, что вы слушаете чаще всего, и рекомендуют пользователю). Нейронные сети, которые получают поступившую от пользователей информацию, анализируют её и прогнозируют то, что может ему понравиться.

Ещё одним направлением применения нейронных сетей является диагностика заболеваний сельскохозяйственных культур. Используя фотографии растений, можно с высокой степенью точности определить вид заболевания и способы его лечения.

Прогнозирование погоды в настоящее время реализовано с помощью приложения «Яндекс.Погода», работающего с высокой точностью и позволяющего определить метеорологические изменения с точностью до нескольких минут.

Распознавание речи по движениям губ человека, основанное на нескольких тысячах видеозаписей, реализовано в английской системе LipNet. Как следует из результатов тестирования, нейронная сеть способна правильно распознавать речь в более чем 90 процентах случаев.

Построение оптимального маршрута движения транспорта в городах, позволяющего объезжать пробки. Примером может служить система Яндекс.Карты.

Можно привести ещё и другие примеры.

В настоящее время искусственный интеллект получает всё большее распространение в сфере образования. Формируются разные подходы по структуризации направлений и областей применения ИИ. Во многих работах высказывается тезис о том, что ИИ — основа современной системы образования. Особое значение ИИ имеет для системы электронного образования. Например, «Акселератор онлайн-школ ACCEL» выделяет следующие направления использования ИИ:

— ИИ становится полноценным участником учебного процесса, обеспечивающим адаптивное и персонализированное обучение и контроль знаний в режиме реального времени. На основе анализа деятельности ученика и учителя в процессе проведения занятий ИИ способен изменить траекторию обучения;
— ИИ позволяет, используя геймификацию, повысить заинтересованность учащихся. Могут быть использованы различные, основанные на игровых подходах программы обучения;
— на основе систем ИИ возможна автоматизация информационных процессов в сфере образования. Это может быть отнесено и к формированию программ и методик обучения, активное применение некоторых электронных образовательных ресурсы исключает участие человека: чат-боты отвечают на вопросы, роботы проводят уроки, ассистируют учителю.

Среда программирования для Arduino

Планируемые результаты освоения учебного предмета с описанием универсальных учебных действий, достигаемых обучающимися

Личностные:

— повышение своего образовательного уровня и уровня готовности к продолжению обучения с использованием ИКТ;
— сформированность представлений о мире профессий, связанных с робототехникой, и требованиях, предъявляемых различными востребованными профессиями, такими как инженер-механик, конструктор, архитектор, программист, инженер-конструктор по робототехнике;
— навыки взаимо- и самооценки, навыки рефлексии.

Предметные:

— определять, различать и называть детали конструктора;
— знать принципы действия электронных и электромеханических элементов;
— понимать назначение элементов, их функцию;
— владеть основами разработки алгоритмов и составления программ управления роботом;
— знать правила соединения деталей в единую электрическую цепь;
— понимать написанный программный код управления устройством, вносить незначительные изменения, не затрагивающие структуру программы (например, значения констант) переменных;
— проводить настройку и отладку конструкции робота;
— записывать отлаженный программный код на плату Arduino, наблюдать и анализировать результат работы;
— проходить все этапы проектной деятельности, создавать творческие работы.

Метапредметные:

— перерабатывать полученную информацию: делать выводы в результате совместной работы всего класса, сравнивать и группировать предметы и их образы;
— самостоятельно выделять и формулировать познавательную цель;
— использовать общие приёмы решения задач;
— контролировать и оценивать процесс и результат деятельности;
— выбирать действия в соответствии с поставленной задачей и условиями её реализации;
— умение выполнять учебные действия в устной форме;
— формулировать собственное мнение и позицию;
— осуществлять взаимный контроль.

Формы контроля

— практическая направленность занятий, выполнение законченного практического проекта на каждом занятии;
— аудиторные занятия в малых группах, индивидуальные образовательные траектории;
— самостоятельное выполнение заданий;
— выполнение итогового проекта;
— тестирование, различные формы опроса.

Нормы оценок знаний и умений учащихся по устному итоговому опросу

Примерный список вопросов

1. Микроконтроллер Arduino.
2. Структура и состав Arduino.
3. Основные комплектующие для схем с Arduino.
4. Tinkercad для Arduino.
5. Устройство светодиода Arduino.
6. Написание скетчев в Tinkercad.

Оценка «5» ставится, если учащийся полностью освоил учебный материал; умеет изложить его своими словами; самостоятельно подтверждает ответ конкретными примерами; правильно и обстоятельно отвечает на дополнительные вопросы учителя.

Оценка «4» ставится, если учащийся в основном усвоил учебный материал, допускает незначительные ошибки при его изложении своими словами; подтверждает ответ конкретными примерами; правильно отвечает на дополнительные вопросы учителя.

Оценка «3» ставится, если учащийся не усвоил существенную часть учебного материала; допускает значительные ошибки при его изложении своими словами; затрудняется подтвердить ответ конкретными примерами; слабо отвечает на дополнительные вопросы.

Оценка «2» ставится, если учащийся почти не усвоил учебный материал; не может изложить его своими словами; не может подтвердить ответ конкретными примерами; не отвечает на бóльшую часть дополнительных вопросов учителя.

Проверка и оценка практической работы учащихся

«5» — работа выполнена в заданное время, самостоятельно, с соблюдением технологической последовательности, качественно и творчески;

«4» — работа выполнена в заданное время, самостоятельно, с соблюдением технологической последовательности, при выполнении отдельных операций допущены небольшие отклонения; общий вид изделия аккуратный;

«3» — работа выполнена в заданное время, самостоятельно, с нарушением технологической последовательности, отдельные операции выполнены с отклонением от образца (если не было на то установки); изделие оформлено небрежно или не закончено в срок;

«2» — ученик самостоятельно не справился с работой, технологическая последовательность нарушена, при выполнении операций допущены большие отклонения, изделие оформлено небрежно и имеет незавершенный вид.

Содержание учебного материала

Урок 1. Знакомство с Arduino

Микроконтроллер Arduino; **применение Arduino;** основные комплектующие для схем с **Arduino (провода, светодиоды, резисторы, пьезоэлемет, кнопки и т. д.);** состав платы Arduino.

Урок 2. Основы программирования в Tinkercad для Arduino

Онлайн-сервис Tinkercad, **возможности** Tinkercad, принципы работы в Tinkercad.

Урок 3. Создание первой схемы в Tinkercad

Электронная схема, библиотеки компонентов, параметры компонентов, виртуальные проводники, элементы, стартовые наборы.

Урок 4. Кейс «Светофор»

Схема светофора для синхронизированной регулировки автомобильного и пешеходного перехода. Алгоритмом работы устройств.

Лабораторная работа № 1. Первые шаги в **Tinkercad**
Лабораторная работа № 2. Написание программы для Arduino
Лабораторная работа № 3. Мигающий светодиод
Лабораторная работа № 4. RGB-светодиод
Лабораторная работа № 5. Кнопка — датчик нажатия
Лабораторная работа № 6. Управление сервоприводом
Лабораторная работа № 7. Светофор на Arduino

Планы учебных занятий

Урок 1

Уровень образования: основное общее
Предмет: информатика
Уровень изучения: базовый
Раздел: Среда программирования для Arduino
Тема урока: Знакомство с Arduino. Основные комплектующие для схем с Arduino
Класс: 8
Тип урока: комбинированный

Цель урока: сформировать у обучающихся представление о микроконтроллерах, познакомить с микроконтроллером Arduino.

Планируемые результаты

Предметные: познакомить с микроконтроллером Arduino, областями его **применения; изучить** основные комплектующие для схем **(провода, светодиоды, резисторы, пьезоэлемет, кнопки и т. д.);** состав платы Arduino.

Метапредметные: ставить учебные задачи под руководством учителя, планировать учебную деятельность; оценивать свою деятельность на уроке на основе критериев достижения результата учебно-познавательной задачи урока; наблюдать, сравнивать, делать умозаключения; устанавливать причинно-следственные связи.

Личностные: проявлять мотивацию к изучению нового учебного материала; способность к самооценке на основе критерия успешности учебной деятельности.

Время реализации: 1 академический час.

Оборудование и материалы: компьютер, проектор, интерактивная доска.

Урок 2

Уровень образования: основное общее
Предмет: информатика
Уровень изучения: базовый
Раздел: Среда программирования для Arduino
Тема урока: Основы программирования в Tinkercad для Arduino
Класс: 8
Тип урока: комбинированный

Цель урока: сформировать у обучающихся представление о сервисе Tinkercad, его возможностях для разработчика Arduino.

Планируемые результаты

Предметные: формирование представления о функционале онлайн-среды разработки Tinkercad, возможностях для Arduino, принципах работы в Tinkercad.

Метапредметные: умение самостоятельно определять цели своего обучения, ставить и формулировать для себя новые задачи в учении и познавательной деятельности; формирование и развитие компетентности в области использования информационно-коммуникационных технологий; приведение примеров, подбор аргументов, формули-

рование обоснованных выводов по обоснованию технико-технологического и организационного решения.

Личностные: формирование интереса к изучению раздела «Среда программирования для Arduino»; готовность и способность обучающихся к саморазвитию и личностному самоопределению; сформированность их мотивации к обучению и целенаправленной познавательной деятельности.

Время реализации: 1 академический час.

Оборудование и материалы: компьютер, проектор, интерактивная доска.

Урок 3

Уровень образования: основное общее

Предмет: информатика

Уровень изучения: базовый

Раздел: Среда программирования для Arduino

Тема урока: Создание первой схемы в Tinkercad

Класс: 8

Тип урока: комбинированный

Цель урока: сформировать у обучающихся представление о создание электронной схемы Arduino.

Планируемые результаты

Предметные: получение навыков построения электронных схем Arduino в Tinkercad, библиотек компонентов, параметров компонентов; получение навыков написание программы для Arduino и её редактирование.

Метапредметные: умение соотносить свои действия с планируемыми результатами; осуществлять контроль своей деятельности в процессе достижения результата; определять способы действий в рамках предложенных условий и требований, корректировать свои действия в соответствии с изменяющейся ситуацией; умение оценивать правильность выполнения учебной задачи, собственные возможности её решения; формирование и развитие компетентности в области использования информационно-коммуникационных технологий.

Личностные: формирование интереса к изучению раздела «Среда программирования для Arduino»; готовность и способность обучающихся к саморазвитию и личностному самоопределению; формирование коммуникативной компетентности в общении и сотрудничестве со сверстниками.

Время реализации: 1 академический час.

Оборудование и материалы: компьютер, проектор, интерактивная доска.

Урок 4

Уровень образования: основное общее

Предмет: информатика

Уровень изучения: базовый

Раздел (согласно тематическому планированию): Среда программирования для Arduino

Тема урока (согласно тематическому планированию): Кейс «Светофор».

Класс: 8

Тип урока: комбинированный

Цель урока: познакомить с алгоритмом работы устройства «светофон», разобрать несколько вариантов написания программы реализации **светофора на Arduino.**

Планируемые результаты

Предметные: получение навыков сборки схемы «светофор» Tinkercad, разработка алгоритмов, реализующих работу светофора.

Метапредметные: умение самостоятельно планировать пути достижения целей, в том числе альтернативные, осознанно выбирать наиболее эффективные способы решения учебных и познавательных задач; умение соотносить свои действия с планируемыми результатами, осуществлять контроль своей деятельности, определять способы действий в рамках предложенных условий, корректировать свои действия в соответствии с изменяющейся ситуацией; умение оценивать правильность выполнения учебной задачи.

Личностные: формирование интереса к изучению раздела «Среда программирования для Arduino»; готовность и способность обучающихся к саморазвитию и личностному самоопределению; готовность признавать возможность существования различных точек зрения и права каждого иметь свою; формирование мотивации к обучению и целенаправленной познавательной деятельности.

Время реализации: 1 академический час.

Оборудование и материалы: компьютер, проектор, интерактивная доска.

Описание лабораторных работ

Лабораторная работа № 1. Первые шаги в Tinkercad

Теоретическая часть

С каждым годом робототехника становиться популярнее. Для разработки роботов и простых проектов используется платформа Arduino, но для изучения данной платформы нужен микроконтроллер **Arduino UNO** или **NANO**, резисторы, светодиоды, соединительные провода, макетная плата, сервоприводы, двигатели, дисплей, реле, датчики и сенсоры. Когда нет под рукой Arduino, можно воспользоваться сервисом Tinkercad.

Tinkercad — это онлайн-сервис, который сейчас принадлежит компании Autodesk, который позволяет собирать электрические цепи и программировать Arduino, а также проверять работоспособность, смоделировав процесс. Чтобы начать работу, достаточно зарегистрироваться в Tinkercad. Главным преимуществом программы, по сравнению с другими CAD-редакторами является то, что она представлена в виде онлайн-приложения и не требует установки на компьютер.

Возможности симулятора Tinkercad для разработчика Arduino

- Удобный графический редактор для визуального построения электронных схем
- Предустановленный набор моделей большинства популярных электронных компонентов, отсортированный по типам компонентов
- Симулятор электронных схем, с помощью которого можно подключить созданное виртуальное устройство к виртуальному источнику питания и проследить, как оно будет работать
- Симуляторы датчиков и инструментов внешнего воздействия. Можно менять показания датчиков, следя за тем, как на них реагирует система
- Встроенный редактор Arduino с монитором порта и возможностью пошаговой отладки
- Готовые для развёртывания проекты Arduino со схемами и кодом

Визуальный редактор кода Arduino

Практическая часть

Цель работы: ознакомление с сервисом Tinkercad.

Ход лабораторной работы

1. Регистрация.

Зайдите на сайт: https://www.tinkercad.com/. Выберите «Вход в аккаунт/создание учётной записи». Подтвердив аккаунт по почте, войдите в систему, указав введённые

параметры. В верхнем правом углу вы увидите ссылку в личный кабинет. В режиме редактирования профиля вы сможете поменять свой псевдоним, e-mail, описание, установить фотографию, подключить внешние сервисы.

После этапа регистрации мы попадаем на главную страницу, в левой части которой видим список модулей, которыми можем воспользоваться, а под ними список проектов.

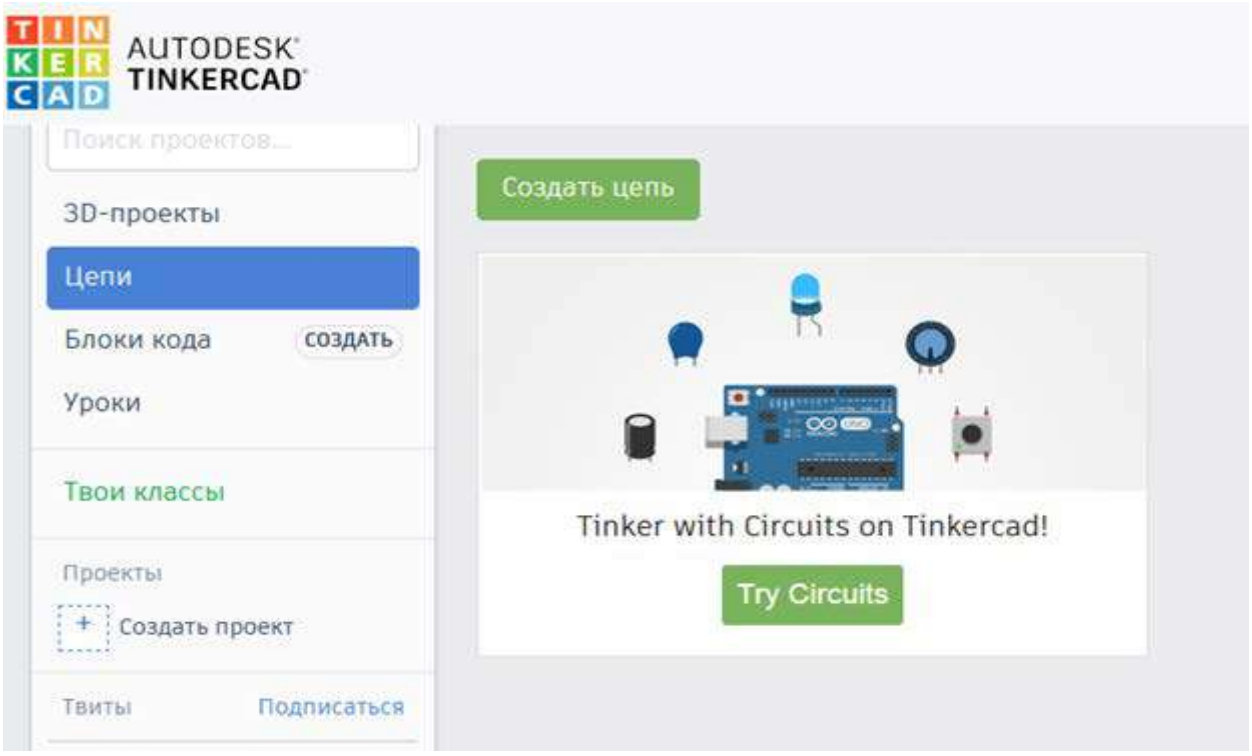

Рис. 161. Вид программы

Чтобы создать новый проект, достаточно нажать кнопку «Создать проект», расположенную под списком проектов. После этого будет создан проект с названием типа Project N. Нажав на него, мы перейдём в режим просмотра списка схем, включённых в этот проект. Там же мы сможем изменить свойства проекта (включая название), нажав на соответствующий значок сразу под названием.

Создать новую схему в Tinkercad можно двумя способами:

1. В меню слева выбрать Circuits и справа над списком схем выбрать команду «Создать цепь». Новая схема будет создана вне какого-либо проекта.

2. Создать схему в определённом проекте. Для этого надо сначала перейти в окно проекта, а затем нажать на кнопку «Создать» сверху над списком. Появится перечень типов схем, мы выбираем «Цепь». Созданная схема будет доступна в этом списке и в списке всех проектов в меню Circuits.

После выполнения команды вы сразу же перейдёте в режим редактирования схемы, не вводя названия. Имя для схемы формируется автоматически.

- Чтобы **изменить название схемы** и отредактировать её свойства, нужно перейти в режим просмотра списка схем, навести на область с названием схемы и нажать на иконку «Настройки». Откроется окно, в котором вы сможете отредактировать параметры.
- Для **удаления** схемы надо в том же режиме выбрать в настройках команду «Удалить».
- Для **просмотра** краткой информации о схеме нужно просто щёлкнуть на неё.

- Для перехода в режим **редактирования** нужно навести курсор мыши и выбрать появившуюся команду «Изменить».

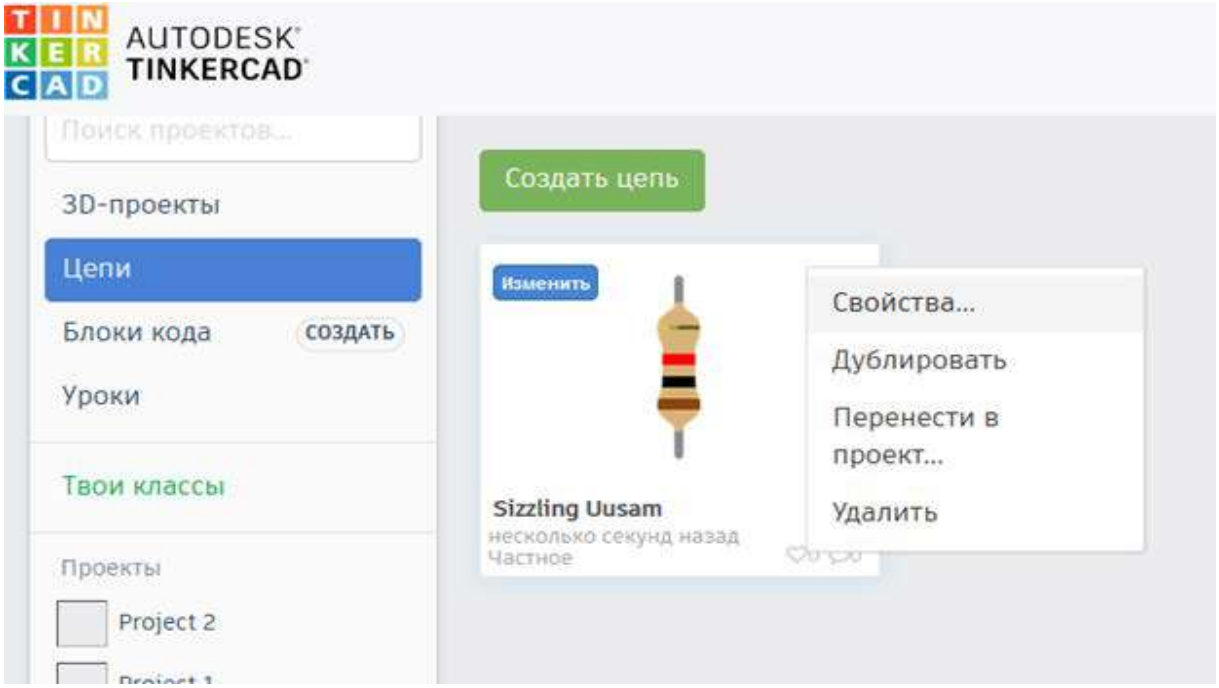

Рис. 162. Вид программы

Нажав на команду «Изменить», мы попадаем в режим редактирования схемы. С помощью удобного и простого графического интерфейса можно нарисовать желаемую электрическую схему. Мы можем выделять, переносить объекты, удалять их привычным всем способом с помощью мыши.

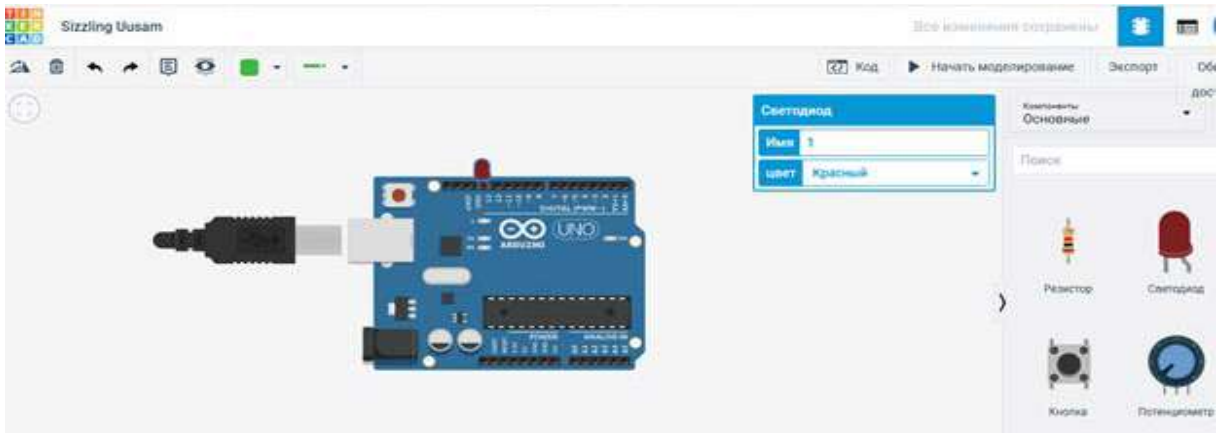

Рис. 163. Вид программы

В режиме редактирования рабочее окно поделено на две части: справа расположена панель с закладками — это библиотека компонентов. Над ней находится область визуального редактирования схемы с панелью инструментов и слева находится область размещения схемы.

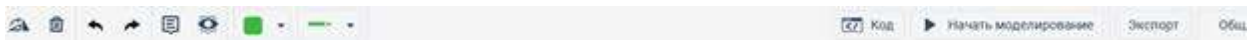

Рис. 164. Панель инструментов

На полосе инструментов в верхней части слева находятся основные команды:

- Повернуть элемент
- Удалить
- Масштабировать по размерам экрана
- Отмена
- Повтор
- Аннотация (создать примечание)
- Скрыть/показать элемент

В правой части:

- Отобразить панель программирования
- Запустить симулятор схемы
- Экспорт
- Поделиться

Составьте схему, представленную на рисунке. Для этого необходимо выбрать светодиод и Arduino Uno.

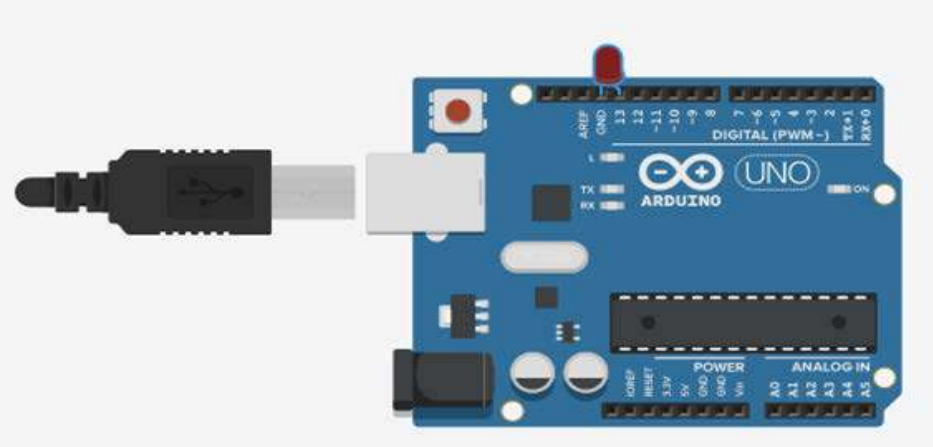

Рис. 165. Вид программы

Запустите проект, выбрав на «Начать моделирование». Загорится светодиод.

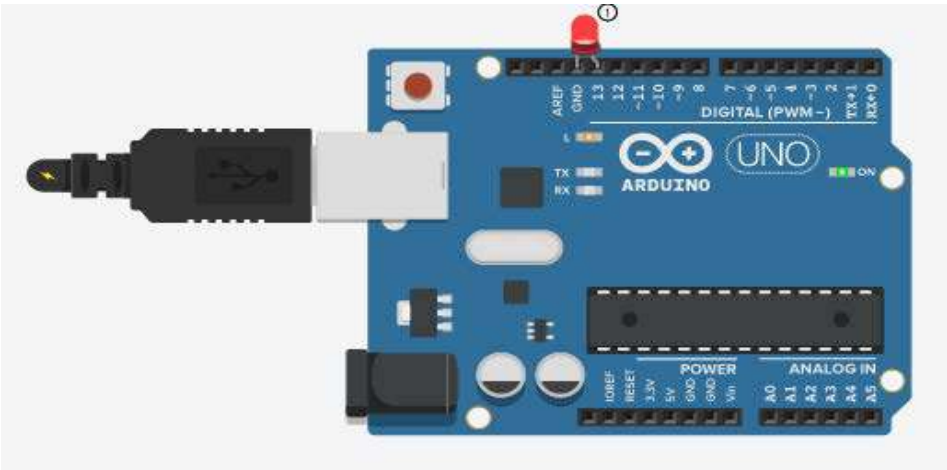

Рис. 166. Вид программы

Выводы: в ходе выполнения лабораторной работы познакомились с симулятором Tinkercad для Arduino.

Контрольные вопросы:

1. Как создать проект в Tinkercad?
2. Перечислите возможности Tinkercad для Arduino.
3. Назовите основные компоненты интерфейса Tinkercad.

Лабораторная работа № 2.
Написание программы для Arduino

Теоретическая часть

В большинстве случае для работы с проектами Arduino выполняется следующий алгоритм действий:

1. Создать новую схему или открыть существующую для редактирования.
2. Используя визуальный редактор, создать схему.
3. Написать скетч в редакторе кода и загрузить его в виртуальный контроллер.
4. Запустить режим симуляции, при которой плата виртуально подключается к источнику питания и схема начинает работать. Внести начальные данные для датчиков и наблюдать реакцию схемы как визуально, так и на виртуальном мониторе порта внутри самого сервиса.

Практическая часть

Цель работы: научиться создавать и программировать схемы в Tinkercad.

Ход лабораторной работы

Создайте новый проект.

Переходим в него и нажимаем на кнопку Create, выбирая тип — Circuit. После этого шага открывается визуальная среда редактирования, в которой мы сможем как нарисовать схему, так и написать и отладить скетч Arduino.

Создавая схему, выполняем такой порядок действий:

1) Выбираем нужные компоненты из библиотеки компонентов внизу экрана и размещаем их в поле редактора.

2) Соединяем компоненты с помощью виртуальных проводников, рисуя их мышкой.

3) Редактируем параметры компонентов (например, величину сопротивления у резисторов или цвет проводов).

Операция выбора из библиотеки достаточно проста. Список элементов находится внизу.

Выбрав элемент, кликаем на него, затем перемещаем в нужное место на схеме и кликаем повторно. Окно со списком компонентов можно скрыть или показать.

Для работы доступно множество готовых элементов, от резистора и батарейки до модулей Arduino. Для удобства навигации все элементы разбиты на три вкладки: Basic

Рис. 167. Вид программы

Components. Основные компоненты; Allcomponents. Все доступные компоненты; Starters. Готовые предустановленные схемы.

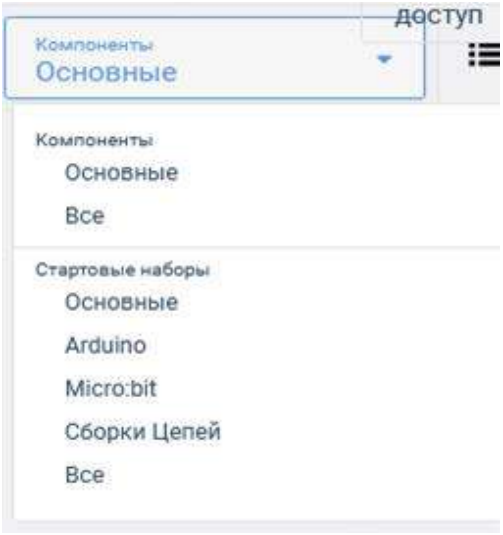

Рис. 168. Вид программы

Выберем закладку Стартовые наборы — Arduino. Создатели сервиса подготовили несколько готовых схем, которые можно сразу же подгрузить в проект и редактировать по своему усмотрению.

Найдите в списке любую схему с Arduino и кликните на неё. После повторного клика элементы схемы будут размещены в области редактирования. Давайте для примера выберем схему трёхкнопочного музыкального инструмента. Разместив её, увидим на экране следующее:

Рис. 169. Вид программы

Для примера выберем схему «Мерцание». Разместив её, мы увидим на экране следующее:

Рис. 170. Вид программы

Кликнув на разъём Arduino или ножки электронных компонентов, можно присоединить к ней провод, который щелчками мышки мы протягиваем по всей нашей схеме до желаемой точки. Для отмены установки провода нужно нажать на Esc или мышкой нажать на соответствующую иконку на панели инструментов.

Все инструменты для редактирования кода становятся доступны после перехода в соответствующий режим при нажатии на кнопку «Код» в верхней панели.

В режиме редактирования кода нам доступны следующие варианты действий:

- загрузить скетч в «виртуальный контроллер» и запустить симулятор;
- переключить в визуальный редактор кода типа Scratch;
- переключить в текстовый редактор кода;
- подключить библиотеки;
- скачать код на свой компьютер в виде файла с расширением .ino (скетч ардуино)
- запустить отладчик с возможностью создания точек остановок и мониторингом состояний переменных
- отобразить или скрыть окно монитора

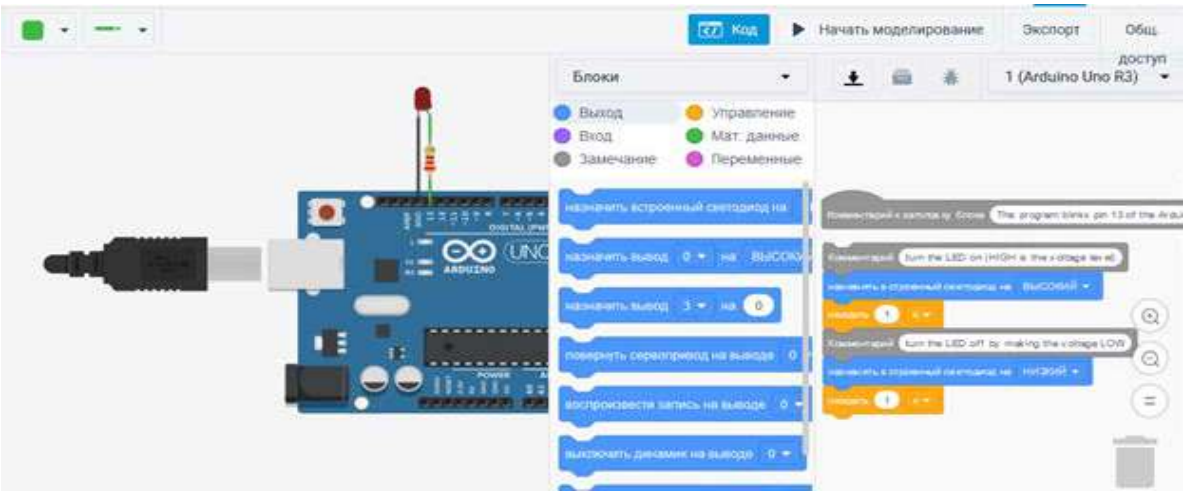

Рис. 171. Вид программы

Для запуска симулятора необходимо нажать на кнопку «Начать моделирование» на верхней панели.

Для остановки нужно нажать на ту же кнопку ещё раз.

Что происходит во время симуляции? Практически то же самое, что и при работе с реальной схемой. Светодиоды светятся, пьезоизлучатель издаёт звуки, валы двигателей вращаются. Мы можем отслеживать текущие показатели (напряжение, ток) с помощью инструментов мониторинга. А также создавать внешние сигналы, подавая на датчики необходимые значения, и отслеживать реакцию программы. Например, можно задать мышкой расположение объекта до датчика расстояния, значение освещённости для фоторезистора, повернуть ручку потенциометра. Также прекрасно работают такие элементы, как LCD-дисплей — выводимая информация отобразится прямо на экране визуального компонента.

Выводы: в ходе выполнения лабораторной работы научились создавать и программировать схемы Arduino в Tinkercad.

Контрольные вопросы:

1. Какие схемы Arduino представлены в Tinkercad?
2. Как запустить процесс моделирования?
3. Как подключить библиотеки в Tinkercad?

Дидактические материалы

Интерактивное задание «Базовые компоненты Arduino»
https://learningapps.org/watch?v=psmvqy3vn21

Рис. 172. Вид Qr-кода

Подпишите основные элементы Arduino.

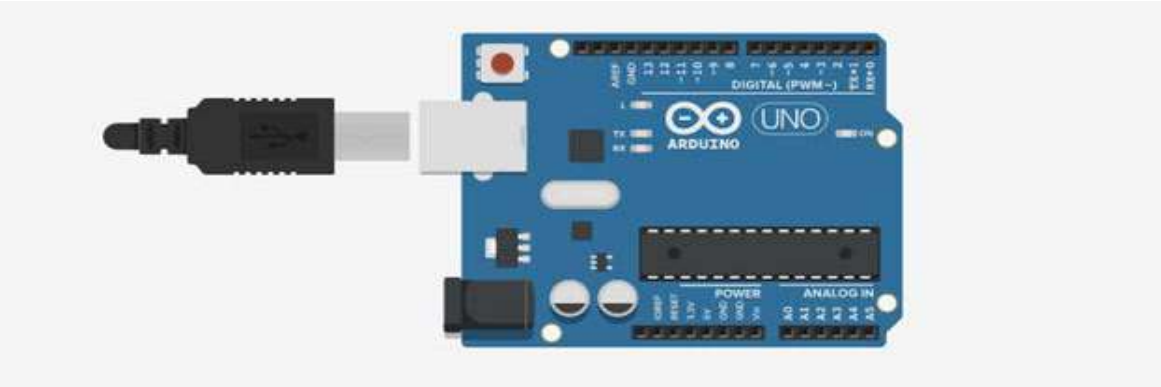

Рис. 172. Иллюстрация к заданию

Технологии кодирования и передачи информации

Планируемые результаты освоения учебного предмета с описанием универсальных учебных действий, достигаемых обучающимися

Личностные:

— формирование умения самостоятельной деятельности;
— формирование умения работать в команде;
— формирование коммуникативных навыков;
— формирование навыков анализа и самоанализа;
— формирование целеустремлённости и усидчивости в процессе творческой, исследовательской работы и учебной деятельности.

Предметные:

— формирование основных понятий, связанных с кодированием и представлением информации;
— формирование понятий о работе с системами счислений;
— формирование знаний об основных приёмах работы в различных позиционных системах счисления;
— формирование знаний об основных способах кодирования различных видов информации.

Метапредметные:

— формирование умения ориентировки в системе знаний;
— формирование умения выбора наиболее эффективных способов решения задач на компьютере в зависимости от конкретных условий;
— формирование приёмов проектной деятельности, включая умения видеть проблему, формулировать тему и цель проекта, составлять план своей деятельности, осуществлять действия по реализации плана, результат своей деятельности соотносить с целью, классифицировать, наблюдать, проводить эксперименты, делать выводы и заключения, доказывать, защищать свои идеи, оценивать результаты своей работы;
— формирование умения распределения времени;
— формирование умений успешной самопрезентации.

Формы контроля

Во время проведения курса предполагается текущий, промежуточный и итоговый контроль.

Текущий контроль осуществляется регулярно во время проведения каждого лабораторного занятия, заключается в ответе учащихся на контрольные вопросы, демонстрации полученных скриптов в среде Scratch, фронтальных опросов учителем.

Также в тематическом планировании предполагается контрольная работа.

Примеры контрольных работы представлены ниже.

1) Переведите в десятичную систему счисления:
а) $10111{,}011_2$
б) $AB3{,}26_{16}$

2) Выполните перевод из десятичной системы счисления:
а) 3428,55 в двоичную
б) 118248 в шестнадцатеричную

3) Вычислите в указанных системах счисления:
а) $1101100000{,}01_2 + 10110110{,}101_2$
б) $1110000110_2 - 101111101_2$
в) $1011001_2 * 1011011_2$
г) $10111_2 : 101_2$
д) $6521_{16} + 3CA8_{16}$
е) $7316_{16} - 6226_{16}$
ж) $694_{16} * AB_{16}$
з) $64B28_{16} : 56_{16}$

Содержание и форма организации учебных занятий

Планы учебных занятий

1. Технологии передачи информации

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение знаний о понятии информации, свойствах информации, технологиях передачи информации.

Метапредметные: умение проанализировать поставленную задачу и те условия, в которых она должна быть реализована, рассматривать разные точки зрения и выбрать правильный путь реализации поставленных задач.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; ответственное отношение к учению, готовность и способность обучающихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию.

2. Кодирование информации

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение знаний о способах кодирования информации, закреплении умения кодировать информацию.

Метапредметные: умение проанализировать поставленную задачу и те условия, в которых она должна быть реализована, рассматривать разные точки зрения и выбрать правильный путь реализации поставленных задач.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; ответственное отношение к учению, готовность и способность обучающихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию.

3. Кодирование числовой информации

Рекомендуемое количество часов на данную тему — 2 часа.

Планируемые результаты

Предметные: получение знаний об определении системы счисления, понятии позиционных и непозиционных системах счисления; основании и алфавите системы счисления; научить переводить числа из десятичной системы счисления в двоичную, восьмеричную и шестнадцатеричную.

Метапредметные: умение проанализировать поставленную задачу и те условия, в которых она должна быть реализована, рассматривать разные точки зрения и выбрать правильный путь реализации поставленных задач.

Личностные: готовность и способность обучающихся к саморазвитию и личностному самоопределению; ответственное отношение к учению, готовность и способность обучающихся к саморазвитию и самообразованию на основе мотивации к обучению и познанию.

Описание лабораторных работ

Лабораторная работа 1. Технологии передачи информации

Теоретический материал

Сигналы и знаки. Человек существует в мире информации, он постоянно получает информацию из окружающего мира с помощью органов чувств, хранит её в своей памяти, анализирует с помощью мышления и обменивается информацией с другими людьми.

Нормальное функционирование живых организмов невозможно без получения и использования информации об окружающей среде. Поведение живых организмов строится на основе получения информационных сигналов. Информационные сигналы могут иметь различную физическую или химическую природу. Это звук, свет, запах и др.

Поведение человека так же, как и животных, строится на основе анализа информационных сигналов, которые он получает с помощью органов чувств. Нервные окончания органов чувств (рецепторы) воспринимают воздействие и передают его по нервной системе в мозг.

Во время общения с другими людьми человек передаёт и получает информацию в форме сообщений с помощью сотен естественных языков. На заре человеческой истории для передачи информации использовался язык жестов, затем появилась устная речь. Человек научился записывать информацию. Первые способы записи информации — это наскальные рисунки. Затем появилась письменность — запись сообщений с помощью специальных знаков — букв.

В соответствии со способом восприятия знаки можно разделить на зрительные, слуховые, осязательные, обонятельные и вкусовые, причём в человеческом общении используются знаки первых трёх типов.

К зрительным знакам, воспринимаемым с помощью зрения, относятся буквы и цифры (они используются в письменной речи), знаки химических элементов, музыкальные ноты, дорожные знаки и т. д.

К слуховым знакам, воспринимаемым с помощью слуха, относятся звуки (они используются в устной речи), а также звуковые сигналы, которые производятся с помощью звонка, колокола, свистка, гудка, сирены и т. д.

Для слепых разработана азбука Брайля, которая использует осязательный способ восприятия текстовой информации. К разряду осязательных знаков принадлежат также жесты-касания: рукопожатия, похлопывания по плечу и др.

В коммуникации многих видов животных особую роль играют обонятельные знаки. Животные помечают свою территорию обитания с помощью запахов, чтобы отпугнуть чужака и показать, что данная территория уже занята.

Для долговременного хранения сообщения с помощью знаков записываются на носители информации.

Знаковые системы. В основе знаковой системы лежит набор знаков, называемый алфавитом. Эти знаки имеют определённую физическую природу. С некоторыми знаковыми системами вы хорошо знакомы и постоянно ими пользуетесь (языки, числа, дорожные знаки), с другими познакомитесь далее. Естественный человеческий язык — это знаковая система. В устной речи знаки — это звуки, в письменной — буквы или алфавит.

В различных науках были разработаны так называемые формальные языки (системы счисления, язык алгебры, языки программирования и др.), отличие которых от естественных языков состоит в существовании ограниченного количества строгих правил грамматики и синтаксиса и в однозначной записи знаками смысла сообщения. Например, десятичную систему счисления можно рассматривать как формальный язык, имеющий алфавит (цифры) и позволяющий именовать и записывать объекты (числа) и выполнять над ними арифметические операции по строго определённым правилам.

Существуют формальные языки, в которых в качестве знаков используют не буквы и цифры, а другие символы, например обозначения химических элементов, музыкальных нот, изображения элементов электрических или логических схем, дорожные знаки, точки и тире (код азбуки Морзе).

Двоичная знаковая система. В процессах хранения, обработки и передачи информации в компьютере используется двоичная знаковая система, алфавит которой состоит всего из двух знаков {0, 1}. Физически знаки реализуются в форме электрических импульсов (нет импульса — 0, есть импульс — 1), а также состояний ячеек оперативной памяти и участков поверхностей носителей информации (одно состояние — 0, другое состояние — 1).

Именно двоичная знаковая система используется в компьютере, так как существующие технические устройства могут надёжно сохранять и распознавать только два различных состояния (знака).

Практическая часть

Цель работы: ознакомление с понятием информации, свойства информации, технологии передачи информации.

Ход лабораторной работы

Работа с программами записи и воспроизведения видео- и аудиофайлов.

Контрольные вопросы:

1. В какой форме в процессе общения с другими людьми человек передаёт и получает информацию?
2. С помощью каких органов человек получает информацию о внешнем мире?
3. Как называется набор знаков, лежащий в основе знаковой системы?
4. В чём состоит отличие формальных языков от естественных?
5. В какой форме реализуются физически знаки двоичной знаковой системы?

Лабораторная работа 2. Кодирование информации

Теоретический материал

Кодирование информации. В процессах восприятия, передачи и хранения информации живыми организмами, человеком и техническими устройствами происходит её кодирование. Результатом кодирования является последовательность знаков данной знаковой системы, т. е. информационный код.

Примерами кодов являются последовательности букв в тексте, цифр в числе, генетический код, двоичный компьютерный код и т. д.

Код состоит из определённого количества знаков (например, текстовое сообщение состоит из определённого количества букв, число — из определённого количества цифр и т. д.), т. е. имеет определённую длину.

Информация, выраженная с помощью естественных и формальных языков в письменной форме, обычно называется текстовой информацией.

Количественная мера информации. Если некоторое сообщение приводит к уменьшению неопределённости нашего знания, то можно говорить, что такое сообщение содержит информацию. Такой подход к информации, как мера уменьшения неопределённости знания, позволяет количественно измерять информацию.

Формула, которая связывает между собой количество возможных информационных сообщений N и количество информации I, которое несёт полученное сообщение, имеет вид:

$$N = 2^I (*).$$

Минимальной единицей измерения количества информации, как вы должны знать из школьного курса информатики, является бит, а следующей по величине единицей — байт, причём:

$$1 \text{ байт} = 8 \text{ бит} = 2^3 \text{ бит}.$$

А кратные байту единицы измерения количества информации вводятся следующим образом:

$$1 \text{ килобайт (Кбайт)} = 2^{10} \text{ байт} = 1024 \text{ байт};$$
$$1 \text{ мегабайт (Мбайт)} = 2^{10} \text{ Кбайт} = 1024 \text{ Кбайт};$$
$$1 \text{ гигабайт (Гбайт)} = 2^{10} \text{ Мбайт} = 1024 \text{ Мбайт}.$$

Формула (*) позволяет легко определить количество информации в сообщении.

Кодирование текстовой информации. Для представления текстовой информации (прописные и строчные буквы русского и латинского алфавитов, цифры, знаки и математические символы) достаточно 256 различных знаков. Можно вычислить, какое количество информации необходимо, чтобы закодировать каждый знак:

$$N = 2^I => 256 = 2^I => 2^8 = 2^I => I = 8 \text{ бит}.$$

Для обработки текстовой информации в компьютере необходимо представить её в двоичной знаковой системе. Для кодирования каждого знака требуется количество информации, равное 8 битам, т. е. длина двоичного кода знака составляет восемь двоичных знаков.

Каждому знаку необходимо поставить в соответствие уникальный двоичный код в интервале от 00000000 до 11111111 (в десятичном коде от 0 до 255).

Человек различает знаки по их начертанию, а компьютер — по их двоичным кодам. При вводе в компьютер текстовой информации происходит её двоичное кодирование. Пользователь нажимает на клавиатуре клавишу со знаком, и в компьютер поступает определённая последовательность из восьми электрических импульсов (двоичный код знака). Код знака хранится в оперативной памяти компьютера, где занимает одну ячейку (размером 1 байт).

В процессе вывода знака на экран компьютера производится обратное кодирование, т. е. преобразование двоичного кода знака в его изображение.

Каждая цифра двоичного компьютерного кода несёт 1 бит информации. Следовательно, сообщение, состоящее из двух цифр, содержит 2 бита информации, из трёх цифр — 3 бита и т. д.

Кодирование графической информации. Графическая информация может быть представлена в аналоговой (непрерывной) или дискретной форме. Примером аналогового представления графической информации может служить живописное полотно, цвет которого изменяется непрерывно, а дискретного — изображение, напечатанное с помощью струйного принтера, состоящее из отдельных точек разного цвета. Графические изо-

бражения из аналоговой формы в цифровую (дискретную) преобразуются путём пространственной дискретизации. Пространственную дискретизацию изображения можно сравнить с построением изображения из мозаики (большого количества маленьких разноцветных стёкол). Изображение разбивается на отдельные маленькие фрагменты (точки, или пиксели), причём каждый элемент имеет свой цвет (красный, зелёный, синий и т. д.).

В результате пространственной дискретизации графическая информация представляется в виде растрового изображения, которое формируется из определённого количества строк, которые, в свою очередь, содержат определённое количество точек.

В процессе дискретизации могут использоваться различные палитры цветов, т. е. наборы тех цветов, которые могут принимать точки изображения. Количество цветов N в палитре и количество информации I, необходимое для кодирования цвета каждой точки, связаны между собой и могут быть вычислены:

$$N = 2^I.$$

В простейшем случае (чёрно-белое изображение без градаций серого цвета) палитра цветов состоит всего из двух цветов (чёрного и белого). Каждая точка экрана может принимать одно из двух состояний («чёрная» или «белая»). Можно вычислить, какое количество информации необходимо, чтобы закодировать цвет каждой точки:

$$2 = 2^I \Rightarrow 2^1 = 2^I \Rightarrow I = 1 \text{ бит}.$$

Количество информации, которое используется для кодирования цвета точки изображения, называется **глубиной цвета**. Наиболее распространёнными значениями глубины цвета при кодировании цветных изображений являются 8, 16 или 24 бита на точку.

Временная дискретизация звука. Для того чтобы компьютер мог обрабатывать звук, непрерывный звуковой сигнал должен быть преобразован в цифровую дискретную форму с помощью временно́й дискретизации. Непрерывная звуковая волна разбивается на отдельные маленькие временны́е участки, причём для каждого такого участка устанавливается определённый уровень громкости. Таким образом, непрерывная зависимость громкости звука от времени заменяется на дискретную последовательность уровней громкости.

Частота дискретизации. Для записи аналогового звука и его преобразования в цифровую форму используется микрофон, подключённый к звуковой плате. Качество полученного цифрового звука зависит от количества измерений громкости звука в единицу времени, т. е. частоты дискретизации. Чем большее количество измерений производится за одну секунду (чем больше частота дискретизации), тем точнее цифровой звуковой сигнал повторяет аналоговый сигнал. Частота дискретизации звука — это количество измерений громкости звука за одну секунду. Частота дискретизации звука измеряется в герцах (Гц) и может лежать в диапазоне от 8000 до 48 000 измерений громкости звука за одну секунду (от 8000 до 48 000 Гц).

Глубина кодирования. Уровни громкости звука можно рассматривать как набор N возможных состояний, для кодирования которых необходимо определённое количество информации I, которое называется глубиной кодирования звука. Глубина кодирования звука — это количество информации, которое необходимо для кодирования дискретных уровней громкости цифрового звука. Если известна глубина кодирования, то количество уровней громкости цифрового звука можно рассчитать.

Пусть глубина кодирования звука составляет 16 бит, тогда количество уровней громкости звука равно:

$$N = 2^I = 2^{16} = 65536.$$

В процессе кодирования каждому уровню громкости звука присваивается свой 16-битовый двоичный код, наименьшему уровню громкости будет соответствовать код 0000000000000000, а наибольшему — 1111111111111111.

Практическая часть

Цель работы: ознакомление с понятием «кодирование», способами кодирования.

Ход лабораторной работы

Задания:

1. Заменяя каждую букву её порядковым номером в алфавите, зашифруйте фразу: «ТИШЕ ЕДЕШЬ — ДАЛЬШЕ БУДЕШЬ».

2. В прямоугольной таблице размером 9 × 9 расположите все буквы алфавита и знаки препинания. Создадим кодировочную таблицу: каждой букве и знаку поставим в соответствие двузначное число — номер строки и номер столбца. Зашифруйте с помощью этой таблицы фразу: «МОСКВА — СТОЛИЦА РОССИИ».

3. Минимальная единица измерения количества информации — это (выберите один правильный вариант):

1) байт 2) бит 3) пиксель 4) дюйм

4. Вычислите, сколько бит содержится в 1 Кбайте, 1 Мбайте, 1 Гбайте.

5. Количество экзаменационных билетов равно 32. Студент берёт экзаменационный билет. Какое количество информации получил студент после того, как увидел номер билета?

6. Известно, что в игре крестики-нолики второй игрок после первого хода первого игрока получил 6 бит информации. Какой размер имеет поле для игры?

7. Какое количество информации несёт сообщение в виде двоичного кода 101010001001?

8. Глубина цвета принимает значения 4, 8, 16, 24, 32 бита. Чему равно количество цветов в палитре для каждого случая?

Контрольные вопросы:

1. Что такое информационный код?
2. Сколько достаточно знаков для кодирования текстовой информации?
3. Какое количество информации необходимо, чтобы закодировать цвет каждой точки?
4. Что такое глубина кодирования звука?

Лабораторная работа 3. Кодирование числовой информации

Теоретический материал

Для записи информации о количестве объектов используются **числа**. Числа записываются с использованием особых знаковых систем, которые называются системами счисления. Алфавит системы счисления состоит из знаков, которые называются цифрами. Система счисления — это знаковая система, в которой числа записываются по определённым правилам с помощью знаков некоторого алфавита, называемых цифрами.

Все системы счисления делятся на две большие группы: позиционные и непозиционные системы. В позиционных системах счисления количественное значение цифры зависит от её положения в числе, а в непозиционных — не зависит.

Непозиционные системы счисления. Как только люди начали считать, у них появилась потребность в записи чисел. Находки археологов на стоянках первобытных людей свидетельствуют о том, что первоначально количество предметов отображали равным количеством каких-либо значков: зарубок, чёрточек, точек. Такая система записи чисел называется единичной, так как любое число в ней образуется путём повторения одного знака, символизирующего единицу.

Примером непозиционной системы, которая сохранилась до наших дней, может служить римская система счисления, которая начала применяться более двух с половиной тысяч лет назад в Древнем Риме.

В основе римской системы счисления лежат знаки I (один палец) для числа 1, V (раскрытая ладонь) для числа 5, X (две сложенные ладони) для 10, а для обозначения чисел 100, 500 и 1000 используются латинские буквы C, D и M.

В римской системе счисления значение цифры не зависит от её положения в числе. Например, в римском числе XXX (30) цифра X встречается трижды и в каждом случае обозначает одну и ту же величину — число 10, три раза по 10 в сумме дают 30.

Чтобы записать число в римской системе счисления, необходимо разложить его на сумму тысяч, полутысяч, сотен, полусотен, десятков, пятёрок, единиц. Например, десятичное число 28 представляется следующим образом: 10 + 10 + 5 + 1 + 1 + 1 = = XXVIII (два десятка, пятёрка, три единицы).

При записи чисел в римской системе счисления применяется правило: каждый меньший знак, поставленный справа от большего, прибавляется к большему знаку, а каждый меньший знак, поставленный слева от большего, вычитается из большего знака.

Например, римское число IX обозначает 9 (−1 + 10), а XI обозначает 11 (10 + 1). Например, число 99 имеет следующее представление в римской системе счисления:

$$XCIX = -10 + 100 - 1 + 10$$

В **позиционных системах счисления** количественное значение цифры зависит от её позиции в числе. Позиция цифры в числе называется **разрядом**. Разряды числа возрастают справа налево, от младших разрядов к старшим, причём значения цифр в соседних разрядах числа различаются в количество раз, равное основанию системы.

В настоящее время наиболее распространённой позиционной системой счисления является десятичная система. В информатике широко используются двоичная, восьмеричная и шестнадцатеричная системы счисления.

В десятичной системе счисления цифра в крайней справа позиции обозначает количество единиц, цифра, смещённая на одну позицию влево, обозначает количество десятков, ещё левее — сотен, затем тысяч и т. д.

Рассмотрим в качестве примера десятичное число 555. Цифра 5 встречается в числе трижды, причём самая правая обозначает пять единиц, вторая справа — пять десятков и, наконец, третья — пять сотен.

Выше десятичное число 555 было записано в привычной для нас свёрнутой форме. Мы настолько привыкли к такой форме записи, что уже не замечаем, как в уме умножаем цифры числа на число 10 в различных степенях, где 10 является основанием десятичной системы счисления.

В развёрнутой форме записи числа умножение цифр числа на основание производится в явной форме. Так, в развёрнутой форме запись числа 555 в десятичной системе будет выглядеть следующим образом:

$$555_{10} = 5 \cdot 10^2 + 5 \cdot 10^1 + 5 \cdot 10^0.$$

Для записи десятичных дробей используются разряды с отрицательными значениями степеней основания. Например, число 555,55 в развёрнутой форме будет записано следующим образом: $555{,}55_{10} = 5 \cdot 10^2 + 5 \cdot 10^1 + 5 \cdot 10^0 + 5 \cdot 10^{-1} + 5 \cdot 10^{-2}$.

Умножение или деление десятичного числа на 10 (величину основания) приводит к перемещению запятой, отделяющей целую часть от дробной, на один разряд соответственно вправо или влево.

Например:

$555{,}55_{10} \cdot 10 = 5555{,}5_{10}$;

$555{,}55_{10} : 10 = 55{,}555_{10}$.

Двоичная система счисления. Числа в двоичной системе в развёрнутой форме записываются в виде суммы основания 2 в различных степенях с коэффициентами, в качестве которых выступают цифры 0 или 1.

Например, развёрнутая запись двоичного числа выглядит следующим образом:

$$A_2 = 1 \cdot 2^2 + 0 \cdot 2^1 + 1 \cdot 2^0 + 0 \cdot 2^{-1} + 1 \cdot 2^{-2},$$

это же число в свёрнутой форме:

$$A_2 = 101{,}01_2.$$

Умножение или деление двоичного числа на 2 (величину основания) приводит к перемещению запятой, отделяющей целую часть от дробной, на один разряд соответственно вправо или влево.

Например:

$101{,}01_2 \cdot 2 = 1010{,}1_2$;

$101{,}01_2 : 2 = 10{,}101_2$.

Восьмеричная система счисления. В восьмеричной системе основание равно 8 и алфавит состоит из восьми цифр {0, 1, 2, 3, 4, 5, 6, 7}. Запишем восьмеричное число в свёрнутой и развёрнутой формах:

$77_8 = 7 \cdot 8^1 + 7 \cdot 8^0$.

Шестнадцатеричная система счисления. В шестнадцатеричной системе основание равно 16 и алфавит состоит из шестнадцати цифр {0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, B, C, D, E, F}, причём первые десять цифр имеют общепринятое обозначение, а для записи остальных цифр со значениями 10, 11, 12, 13, 14, 15 используются первые шесть букв латинского алфавита. Запишем шестнадцатеричное число в свёрнутой и развёрнутой формах:

$$ABCDEF_{16} = A \cdot 16^5 + B \cdot 16^4 + C \cdot 16^3 + D \cdot 16^2 + E \cdot 16^1 + F \cdot 16^0 =$$
$$= 10 \cdot 16^5 + 11 \cdot 16^4 + 12 \cdot 16^3 + 13 \cdot 16^2 + 14 \cdot 16^1 + 15 \cdot 16^0.$$

Первая позиционная система счисления была придумана ещё в Древнем Вавилоне, причём вавилонская нумерация была шестидесятеричной, т. е. в ней использовалось шестьдесят цифр! Интересно, что до сих пор при измерении времени мы используем основание, равное 60 (в 1 минуте содержится 60 секунд, а в 1 часе — 60 минут).

В XIX в. довольно широкое распространение получила двенадцатеричная система счисления. До сих пор мы часто употребляем дюжину (число 12): в сутках две дюжины часов, круг содержит тридцать дюжин градусов и т. д.

Практическая часть

Цель работы: ознакомление с основными понятиями позиционных систем счисления, получения навыков по работе в различных позиционных системах счислениях.

Ход лабораторной работы

1. Перевести из 2сс в 10сс

1) 10101011
2) 11001110

2. Перевести из 10сс в 2сс

1) 3524
2) 345,5

3. Выполнить действия в 2сс

1) Сложение

101010 + 1101

10000111 + 101010

11110001 + 11100

2) Вычитание
   101010111 — 1111
   1100011 — 1011
   10100001 — 1111
3) Умножение
   1010 · 11
   110011 · 101
   1100 · 101
4) Деление
   1111 : 11
   11110 : 101
   110110 : 110

Контрольные вопросы:

1. Что такое система счисления?
2. В чём состоит отличие позиционной системы счисления от непозиционной? Приведите примеры таких систем.
3. Какая система счисления используется человеком в повседневной жизни? Какая используется в компьютере?

Дидактические материалы

Пример кодирования букв русского алфавита с помощью кода Шеннона—Фано

Буква	Код	Буква	Код
пробел	0	я	110110
о	001	ы	110111
е	0100	з	111000
а	0101	ъ,ь	111001
и	0110	б	111010
т	0111	г	111011
н	1000	ч	111100
с	1001	й	1111010
р	10100	х	1111011
в	10101	ж	1111100
л	10110	ю	1111101
к	10111	ш	11111100
м	11000	ц	11111101
д	110010	щ	11111110
п	110011	э	111111110
у	11010	ф	111111111

Материалы для организации и проведения учебно-исследовательской и проектной деятельности школьников

Мероприятие «День открытых дверей»

Цель: популяризация деятельности центра «Точка роста» в игровой форме (игра по станциям).

Задача каждой команды — пройти предложенные испытания на каждой образовательной Точке, перечисленные в маршрутном листе:

Точка «Программирование на Python»
Точка «Технологии кодирования и передачи информации»
Точка «Среда программирования Scratch»
Точка «Среда программирования для Arduino»
Точка «Робототехника»
Точка «Вопросы искусственного интеллекта».

На выполнение каждого задания отводится 6 минут, по истечении времени команда переходит к следующей Точке роста. Педагоги будут строго отслеживать время и качество выполнения задания. В маршрутном листе команды каждый педагог в соответствии с определёнными критериями будет оценивать вашу работу. Пройдя маршрут всех образовательных Точек, мы собираемся в этом кабинете для подведения итогов и проведения рефлексии мероприятия.

Описание заданий образовательных точек игры

Точка «Программирование на Python»

Задание. Попробуем нарисовать простую картинку из символов: буквы «о», знака подчёркивания и вертикальной черты. Напишите следующий код:

```
print (‘Привет!’)
print (‘Это изображение собаки’)
print (‘o_______’)
print (‘| | | |’)
```

А теперь запустите выполнение программы с помощью кнопки Run. У вас должен получиться результат «собачка». Необязательно выводить каждую строку изображения с помощью отдельной функции print(). Можно вывести несколько строк с помощью однократного вызова функции, для этого заключите весь текст в тройные кавычки.

А теперь попробуйте создать свой собственный рисунок из символов. Вы можете создать целый сюжет или открытку к празднику. Используйте точки, скобки, знаки подчёркивания и любые другие символы.

Внимание! Не используйте верхние кавычки в процессе «рисования» символами — три кавычки подряд «закроют» текст. Также стоит быть осторожнее с двоеточиями — об их роли в Python вы узнаете чуть позже.

Точка «Технологии кодирования и передачи информации»

Задание 1. «Пароль»

Координатное устройство для управления курсором и отдачи различных команд компьютеру. О каком устройстве идёт речь? Устройство — это пароль. На координатной плоскости отметьте точки, координаты которых приведены ниже. Соедините точки по возрастанию. Помните, первое число — координата по оси OX, второе число — координата по оси OY. Полученная картинка — это пароль.

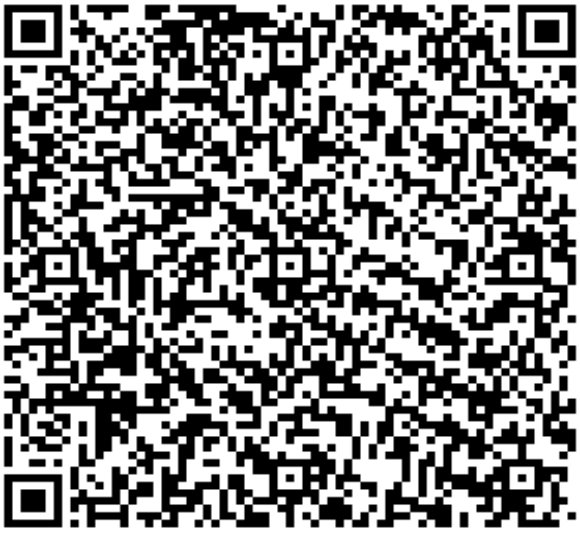

Рис. 173. Вид Qr-кода

Задание 2. «Головоломка»

Расшифруйте известных специалистов в области информатики и информационных технологий (карточки с QR-кодами распечатать заранее). Найдите:

		
		

1-я пара — создатели одного из языков программирования.
2-я пара — создатели массовых компьютеров компании Apple.
3-я пара — создатели социальных сетей.

Точка «Среда программирования Scratch»

Задание. Найдите зашифрованные понятия Scratch.

Интерактивное задание в learningapps.org.

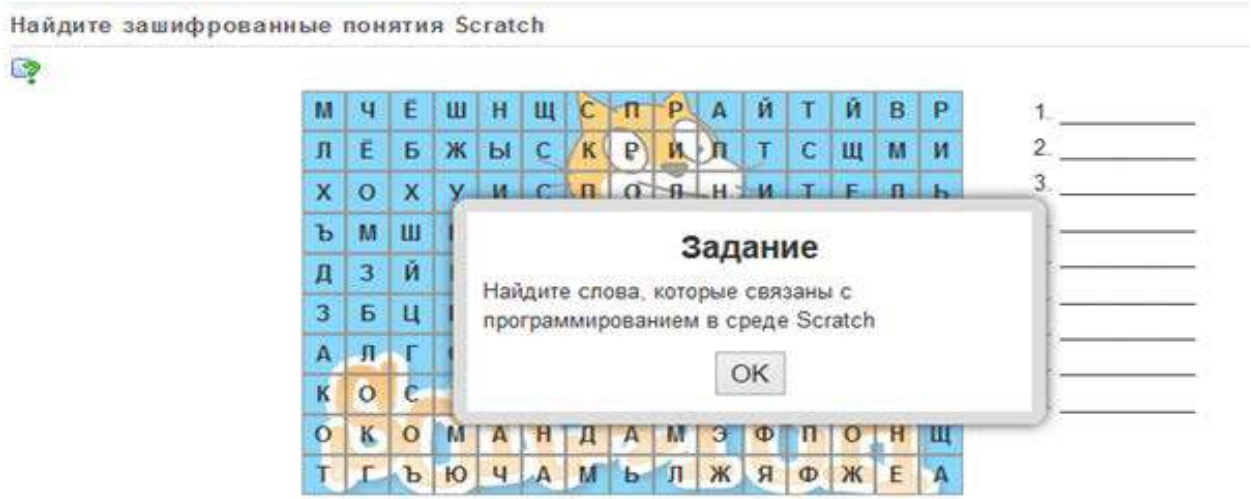

Рис. 174. Вид задания

Рис. 175. Вид Qr-кода

Точка «Среда программирования для Arduino»

Задача. Собрать схему по образцу и научится управлять светодиодом (программирование миганием светодиода).

Порядок подключения:

1. Длинную ножку светодиода (анод) подключаем к цифровому выводу D10 Arduino, другую (катод) — через резистор 220 Ом к выводу GND.
2. Загружаем в плату Arduino скетч из листинга.
3. Наблюдаем процесс мигания светодиода.

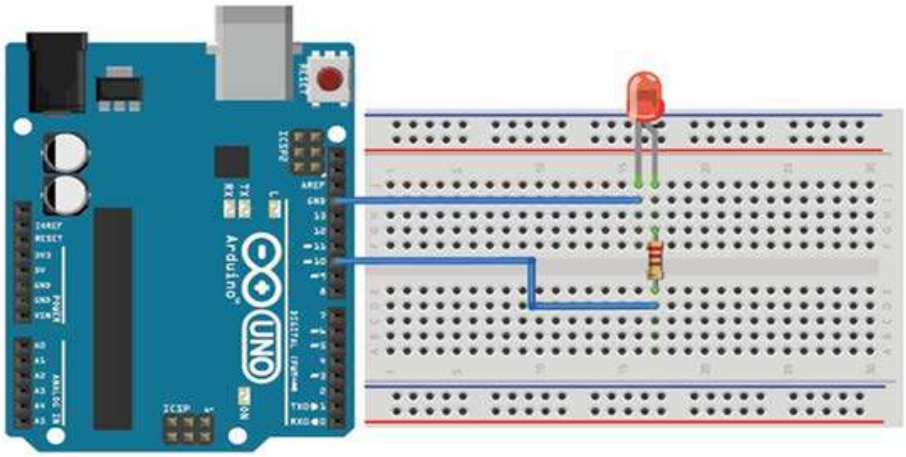

Рис. 176. Вид задания

Скетч эксперимента приведён в листинге.

```
const int LED=10; // вывод для подключения светодиода 10 (D10)
void setup()
{
// Конфигурируем вывод подключения светодиода как выход (OUTPUT)
pinMode(LED, OUTPUT);
}
void loop()
{
// включаем светодиод, подавая на вывод 1 (HIGH)
digitalWrite(LED,HIGH);
// пауза 1 сек (1000 мс)
delay(1000);
// выключаем светодиод, подавая на вывод 0 (LOW)
digitalWrite(LED,LOW);
// пауза 1 сек (1000 мс)
delay(1000);
}
```

Теперь мы можем поэкспериментировать с периодом мигания светодиода, меняя в скетче значения задержки в функции delay().

Точка «Робототехника»

Прохождение интерактивного «Робоквеста».

https://docs.google.com/presentation/d/19GXlq_k9rTQ4FsurbcbowTNl6pjsuG6710So-9blFt0/edit?usp=sharing

Рис. 177. Вид задания

Рис. 178. Вид Qr-кода

Точка «Вопросы искусственного интеллекта».
https://infourok.ru/metodicheskaya-razrabotka-uroka-tema-iskusstvenniy-intellekt-3112186.html

Соревнования по робототехнике

Цель конкурса — привлечение учащихся к инновационному, научно-техническому творчеству в области робототехники.

В соревнованиях принимают участие учащиеся от 7 лет. Конкретные возрастные ограничения оговариваются для каждой номинации.

Номинация «Юный роботехник».

В состязаниях участвуют учащиеся 7—9 лет. Соревнования в данной номинации проводятся по индивидуальной системе (один участник — один конструктор). Участники должны собрать модель по опорной схеме и составить для неё программу по предложенному заданию и объяснить её.

Победителем в номинации объявляется участник, набравший наибольшее количество баллов (максимальное количество баллов — 50).

1) сборка робота согласно схеме — до 20 баллов;
2) программирование — до 20 баллов;
3) презентация модели — до 10 баллов.

Номинация «Робо-поло».

В состязании участвуют учащиеся 9—14 лет. Игровое поле для «Поло» имеет размеры 2000 × 1100 мм белого цвета.

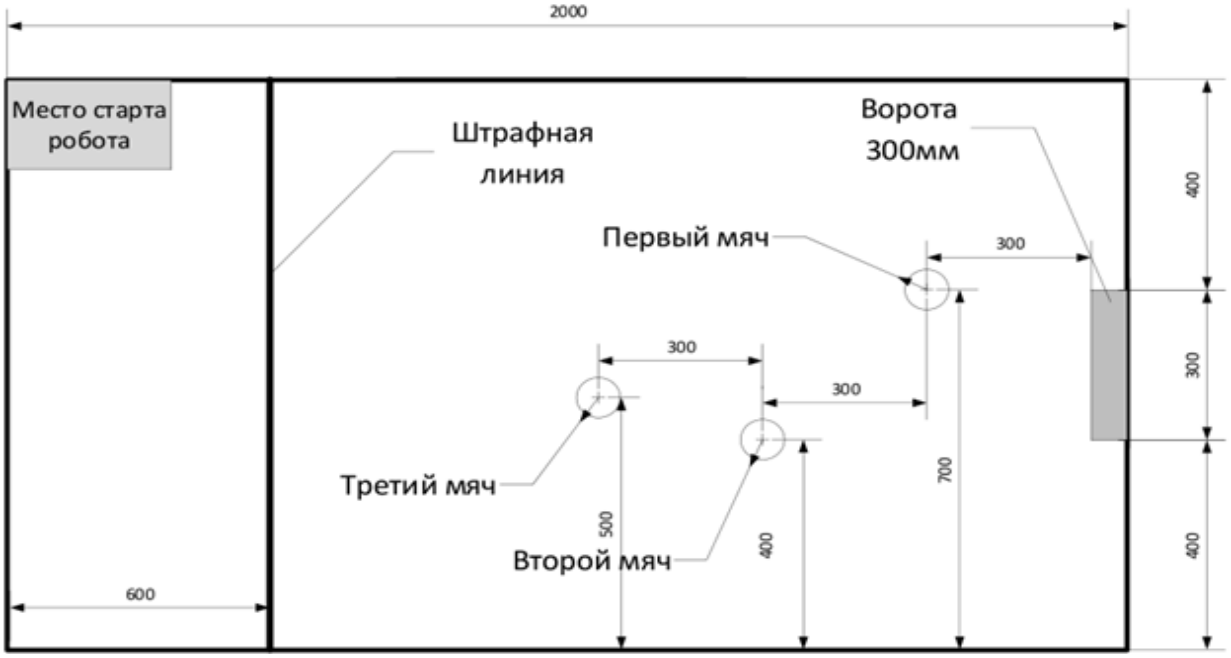

Рис. 179. Вид поля

Воротами являются две горизонтальные планки, ширина створа ворот 300 мм. В качестве мячей выступают теннисные шарики. Места расположения мячей обозначены на рисунке. Мячи устанавливаются на подставки произвольной формы высотой 10—15 мм и должны быть промаркированы цифрами маркером.

Роботы должны быть построены только из деталей набора LEGO. Роботы будут измеряться в вертикальном положении, при этом они не должны ни на что опираться и их под-

вижные части должны быть максимально выдвинуты. Высота робота должна составлять не более 350 мм.

Состязание состоит из индивидуальных матчей, каждый из которых длится до трёх забитых мячей в ворота, но не более четырёх минут (мяч представляет собой шарик для игры в настольный теннис). Порядок выступления участников определяется по номеру регистрации.

Судьи фиксируют время каждого забитого мяча и заносит его в таблицу. Лучшим считается игрок, затративший наименьшее время на все забитые мячи, если игроки показали одинаковое время по трём забитым мячам, то победителем является игрок показавший лучшее время по второму забитому мячу и т. д.

Робот изначально находится в исходной точке, обозначенной на поле. Задача игрока: управляя роботом, забить мячи в створ ворот. Последовательность забиваемых мячей определяет сам участник. В случае, если мяч покидает границы игровой зоны, не попав в створ ворот, робот должен вернуться за границу штрафной линии, в это время помощник возвращает мяч на подставку в обозначенное место, откуда мяч был перемещён.

По итогам первого тура составляется рейтинг команд на основании количества забитых мячей. Лучшие четыре команды выходят в 1/4 финала. В случае опрокидывания робота и в течение 30 секунд невозможности продолжить игру участник выбывает с нулевым результатом.

Матчи будут проходить по тем же правилам. Победители будут определяться из четырёх команд.

Номинация «Решение задач Arduino».

Участникам необходимо за 90 минут собрать различные схемы на базе платформы Arduino и запрограммировать их. Задания выполняются в среде моделирования TinkerCAD в соответствии с инструкцией, полученной в день проведения конкурса от организаторов. Баллы за задание начисляются только при его 100%-м выполнении.

Пример заданий.

1. Подключить три светодиода (красный, жёлтый, зелёный) и смоделировать работу светофора: 3 секунды горит только красный светодиод, 2 секунды только мигающий жёлтый (0,5 секунды горит, 0.5 секунды не горит, повтор 2 раза), 2 секунды горит только зелёный.

2. Подключить датчик газа и пьезоэлемент. При значении с датчика больше 500 раздаётся звук с частотой 500 Гц, при значении меньше 500 — звук не воспроизводится.

3. Подключить эластичную клавиатуру и 7-мисегментный индикатор. При нажатии цифры «0» на индикаторе появляется «0», при нажатии «1» — «1» и т. д. до 9. Символы и буквы выводить не нужно.

Хакатон по программированию

Хакатон — соревновательное мероприятие (конкурс) для школьников, на котором начинающие IT-специалисты разрабатывают прототип будущих приложений в образовательном сегменте с использованием средств информационных технологий.

Целью хакатона являлась популяризация языка программирования на Python и Scratch, развитие научно-технического творчества среди обучающихся.

В хакатоне участвуют команды, в состав которых могут входить не менее двух и не более четырёх обучающихся 2—7 классов от одной образовательной организации под руководством наставника. Команды формируются на основании возраста участников в группы от 8 до 11 и от 11 до 14 лет. Участники одной возрастной группы должны обладать навыками программирования на Scratch или Python.

Тема хакатона — «Совершенствование образовательной среды».

Задача хакатона: определить проблему, соответствующую теме хакатона, создать продукт (приложение, игра, сайт, презентация, анимация), который будет демонстрировать решение этой проблемы.

Работа над проектами состоит из двух этапов. Результатом первого этапа является план работы над проектом. Результат второго этапа — продукт или прототип продукта.

После завершения второго этапа команды демонстрируют работоспособность продукта (регламент выступления — не более 7 минут: 3 минуты на выступление, 2 минуты на демонстрацию прототипа и 2 минуты — ответы на вопросы).

Оценка проектов производится по четырём основным направлениям;

— идея;
— продукт (функциональность, техническое исполнение);
— менеджмент (распределение ролей, соблюдение тайминга);
— дизайн (узнаваемость, качество исполнения);
— выступление (полнота раскрытия темы, качество выступления).

Победителями конкурса признаются команды, а также отдельные лица (номинанты), набравшие наибольшее количество баллов в соответствии с критериями оценивания. Каждый пункт оценивается по 10-балльной системе.

Соревнования по программированию на Scratch

Конкурс проводится по номинациям: игровое приложение; социальный мультфильм; приложение, позволяющее на основе данных, введённых пользователем, решить оптимизационную задачу; анимация статичных, рисованных рисунков или фотографий.

Работа должна содержать: законченное приложение или обучающий мультфильм с возможностью просмотра исходного кода.

Работы оцениваются по следующим критериям (примерный список):

— актуальность и практическая значимость разработки;
— работоспособность (программа работает без сбоев);
— насыщенность элементами мультимедийности;
— наличие анимации объектов;
— понятный и законченный сюжет истории;
— наличие титров;
— сложность программирования;
— креативность к подходу (создание собственных спрайтов, фонов и т. д.);
— соответствие теме соревнований.

Кейс «Знайки». Необходимо подготовить фон и героев для проекта. Участник выбирает одну из предметных/межпредметных областей: «Математика», «История», «Робототехника» и т. д., придумывает сюжет, создаёт персонажи, выполняющие роль ведущих викторины, составляет разные типы вопросов, программирует счётчик правильных/неправильных ответов. Необходимо продумать звуковое сопровождение к проверке ответа игрока (например, аплодисменты).

Мастер-класс «Создай свой виртуальный тур»

Цель — познакомить с понятием «виртуальный тур», областью применения 3D-панорамы, выделить достоинства и недостатки панорамам 360°. В ходе мастер-класса участники разработают простой 3D-тур от первого лица с помощью сервиса Tour Creator (https://arvr.google.com/tourcreator/) на основе Google.

Оборудование: VR-очки (шлем виртуальной реальности), проектор, интерактивная доска, компьютер.

Мастер-класс «Google Cardboard или VR-очки своими руками»

Цель — знакомство с понятием «виртуальная реальность», исследование возможностей различных VR-устройств. В ходе мастер-класса участники выполняют проектную задачу — научатся конструировать модель VR-очков из выбранного материала.

Образовательный интенсив «3D-моделирование и 3D-печать»

Цель — познакомить участников с виртуальной реальностью, дать основные навыки работы с инструментарием виртуальной реальности, обучение участников актуальным технологиям современного цифрового производства, навыкам 3D-моделирования, прототипирования, 3D-сканирования, 3D-печати.

На первом этапе проведения интенсива проходит погружение в 3D-моделирование: 3D-технологии (современное состояние, перспективы, AR\VR, 3D-оборудование), работа с 3D-графикой, особенности печати 3D-моделей.

На втором этапе участники в соответствии со своими интересами записываются на проекты.

Итогом интенсива будет представление (защита проектов) 3D-модели, включающей элементы, напечатанные на 3D-принтере и просмотренные в виртуальной реальности.

Перечень доступных источников информации

1. https://scratch.mit.edu/ Сообщество Sctach.
2. Python для начинающих 2021 — уроки, задачи и тесты https://pythonru.com/uroki/python-dlja-nachinajushhih
3. Python/Учебник Python 3.1 https://ru.wikibooks.org/wiki/Python/%D0%A3%D1%87%D0%B5%D0%B1%D0%BD%D0%B8%D0%BA_Python_3.1
4. *Босова Л. Л.* Информатика. 8 класс: учебник. — М.: БИНОМ. Лаборатория знаний, 2016. — 176 с.
5. *Буйначев С. К.* Основы программирования на языке Python: учеб. пособие. — Екатеринбург: Изд-во Урал. ун-та, 2014. — 91 с.
6. *Бхаргава А.* Грокаем алгоритмы. Иллюстрированное пособие для программистов и любопытствующих. — СПб.: Питер, 2017. — 288 с.
7. *Бэрри П.* Изучаем программирование на Python. — М., 2017. — 624 с.
8. *Винницкий Ю. А.* Scratch и Arduino для юных программистов и конструкторов. — СПб: БХВ-Петербург, 2018. — 176 с.
9. *Голиков Д. В.* Scratch для юных программистов. — СПб.: БХВ-Петербург, 2017. — 192 с.
10. *Гэддис Т.* Начинаем программировать на Python / Пер. с англ. — 4-е изд. — СПб.: БХВ-Петербург, 2019. — 768 с.
11. Лаборатория юного линуксоида. Введение в Scratch. http://younglinux.info/scratch
12. *Луридас П.* Алгоритмы для начинающих: теория и практика для разработчика. — М. : Эксмо, 2018. — 608 с.
13. *Лутц М.* Изучаем Python / Пер. с англ. — 3-е изд — СПб.: Символ Плюс, 2009. — 848 с.
14. *Маржи М.* Scratch для детей. Самоучитель по программированию — пер. с англ. М. Гескиной и С. Таскаевой. — М. : Манн, Иванов и Фербер, 2017. — 288 с.
15. *Мюллер Дж.* Python для чайников. — СПб.: Диалектика, 2019. — 416 с.
16. *Пашковская Ю. В.* Творческие задания в среде Scratch. Рабочая тетрадь для 5—6 классов. — М., 2018. — 195 с.
17. *Первин Ю. А.* Методика раннего обучения информатике. — М.: «Бином», Лаборатория базовых знаний, 2008. — 228 с.
18. *Поляков К. Ю.* Информатика. 7 класс (в 2 частях) : учебник. Ч. 1 / К. Ю. Поляков, Е. А. Еремин. — М.: БИНОМ. Лаборатория знаний, 2019. — 160 с.
19. Практический Python 3 для начинающих https://pythonworld.ru/samouchitel-python.
20. *Рафгарден Т.* Совершенный алгоритм. Жадные алгоритмы и динамическое программирование. — СПб.: Питер, 2020. — 256 с.
21. *Рейтц К., Шлюссер Т.* Автостопом по Python. — СПб.: Питер, 2017. — 336 с.
22. *Рындак В. Г., Дженжер В. О., Денисова Л. В.* Проектная деятельность школьника в среде программирования Scratch: учебно-метод. пособие. — Оренбург: Оренб. гос. ин-т менеджмента, 2009. — 116 с.
23. *Свейгарт Эл.* Программирование для детей. Делай игры и учи язык Scratch!. — М.: Эксмо, 2017. — 304 с.
24. *Семакин И. Г., Залогова, Л. А. и др.* Информатика и ИКТ: учебник для 9 класса. — М.: Бином, 2014. — 171 с.
25. *Торгашева Ю.* Первая книга юного программиста. Учимся писать программы на Scratch. — СПб.: Питер, 2016. — 128 с.

26. *Уфимцева П. Е., Рожина И. В.* Обучение программированию младших школьников в системе дополнительного образования с использованием среды разработки Scratch // Наука и перспективы. — 2018. — № 1. — с. 29—35.

27. Учебник по языку программирования Python (хабраиндекс) https://habr.com/ru/post/61905/

28. *Федоров Д. Ю.* Программирование на языке высокого уровня Python: учеб. пособие для прикладного бакалавриата. — М. : Издательство Юрайт, 2019. — 161 с.

29. *Адаменко А. Н., Кучуков А. М.* Логическое программирование и Visual Prolog. — СПб.: БХВ-Петербург, 2003. — 992 с.

30. *Братко И.* Программирование на языке Visual Prolog для искусственного интеллекта. — М.: Мир, 1990. — 560 с.

31. *Ин Ц., Соломон Д.* Использование Турбо—Пролог. — М.: Мир, 1993. — 608 с.

32. *Стерлинг Л., Шапиро Э.* Искусство программирования на языке Visual Prolog. — М.: Мир, 1990. — 235 с.

Григорьев Сергей Георгиевич
Востроknутов Игорь Евгеньевич
Родионов Михаил Алексеевич
Акимова Ирина Викторовна
Кочеткова Ольга Анатольевна

Реализация образовательных программ по предмету "Информатика" с использованием оборудования центра «Точка роста»

Методическое пособие

Под редакцией С. Г. Григорьева

Центр Естественно-научного и математического образования
Руководитель Центра *З. Г. Гапонюк*
Ответственный за выпуск *Е. С. Карауш*
Редактор *Е. С. Карауш*
Художественное оформление *С. И. Ситников*
Компьютерная вёрстка и техническое редактирование *О. Ю. Мызникова*
Корректор *Г. И. Мосякина*