МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ЮУрГГПУ»)

ФАКУЛЬТЕТ МАТЕМАТИКИ, ФИЗИКИ, ИНФОРМАТИКИ

КАФЕДРА ИНФОРМАТИКИ, ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И
МЕТОДИКИ ОБУЧЕНИЯ ИНФОРМАТИКЕ

Методика обучения языку программирования Python в системе дополнительного образования

Выпускная квалификационная работа по направлению

44.04.01 Педагогическое образование

Направленность программы магистратуры

«Информатика и робототехника в образовании»

Форма обучения заочная

Проверка на объем заимствований:
79,46 % авторского текста
Работа рекомендована к защите
рекомендована/не рекомендована
« 12 » февраля 2024 г.
зав. кафедрой
ИИТиМОИ ЮУрГГПУ
Рузаков Андрей Александрович

Выполнил:
Студент группы ЗФ-313-276-2-1
Сайдуллина Екатерина Вячеславовна

Научный руководитель:
кандидат педагогических наук, доцент,
доцент кафедры ИИТиМОИ ЮУрГГПУ
Поднебесова Галина Борисовна

Челябинск

2024

МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«ЮЖНО-УРАЛЬСКИЙ ГОСУДАРСТВЕННЫЙ
ГУМАНИТАРНО-ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ»
(ФГБОУ ВО «ЮУрГГПУ»)

ФАКУЛЬТЕТ МАТЕМАТИКИ, ФИЗИКИ, ИНФОРМАТИКИ

КАФЕДРА ИНФОРМАТИКИ, ИНФОРМАЦИОННЫХ ТЕХНОЛОГИЙ И
МЕТОДИКИ ОБУЧЕНИЯ ИНФОРМАТИКЕ

Методика обучения языку программирования Python в системе дополнительного образования

Выпускная квалификационная работа по направлению

44.04.01 Педагогическое образование

Направленность программы магистратуры

«Информатика и робототехника в образовании»

Форма обучения заочная

Проверка на объем заимствований:
_____% авторского текста
Работа _____к защите
рекомендована/не рекомендована
« _____» _____20___г.
зав. кафедрой
ИИТиМОИ ЮУрГГПУ
Рузаков Андрей Александрович

Выполнил:
Студент группы ЗФ-313-276-2-1
Сайдуллина Екатерина Вячеславовна

Научный руководитель:
кандидат педагогических наук, доцент,
доцент кафедры ИИТиМОИ ЮУрГГПУ
Поднебесова Галина Борисовна

Челябинск

2024

Содержание

ВВЕДЕНИЕ

Развитие информационных технологий требует постоянного прилива специалистов для их обслуживания и совершенствования. Однако, несмотря на различные усилия государства, ощущается постоянный недостаток специалистов в этой области. Частично это связано с повышающимся, но еще достаточно низким запросом в обществе на получение такого рода знаний.

Кроме этого, реальная сложность в изучении программирования накладывается на низкий уровень подготовки школьников. Статистика по числу участников в государственной итоговой аттестации по информатике в рамках ЕГЭ и ОГЭ за последние годы говорит о повышении интереса к сдаче экзамена по информатике, что отразилось в постоянном росте числа сдающих.

Однако уровень знаний школьников не растет на волне интереса к информационным технологиям. Большинство воспринимают информатику как предмет, в рамках которого они получают необходимые пользовательские функции. Остальные аспекты образовательного процессы не вызывают интереса у школьников, поэтому малая часть занимается изучением программирования.

Низкий уровень мотивации к обучению является серьезным препятствием в получении прочных знаний и навыков, а, следовательно, ведет к не очень хорошим результатам в ходе оценки знаний посредством основного и единого государственных экзаменов по информатике.

Поиск путей повышения уровня мотивации к обучению в таком сложном предмете как информатика является очень актуальным в связи с необходимостью увеличения числа специалистов в области информационных технологий.

Возникает серьезное противоречие между реальным интересом

среди школьников к информационным технологиям в целом, и при этом низким уровнем познавательной деятельности при изучении, например, программирования. Таким образом «мотивация» и ее регулирование становится важным направлением деятельности преподавателя информатики.

Анализ существующих технологий повышения мотивации, используемых в рамках других дисциплин, не в полной мере отвечает потребностям информатики как предмета изучения. Несмотря на практическую реализацию информационных технологий в рамках учебной деятельности используются искусственные математические задачи для обучения программированию. В таких условиях возникает быстрая потеря интереса к изучению программирования, так как школьник не видит применения полученных знаний.

Практическая направленность решаемых задач, а также выбор индивидуальных траекторий обучения для разных обучаемых мог бы в должной мере повысить уровень способных к программированию школьников и поддержать их познавательный интерес.

Таким образом, была сформулирована **тема исследования**: методика обучения языку программирования Python в системе дополнительного образования.

Цель исследования: теоретически обосновать и экспериментально проверить эффективность методики обучения программированию школьников в системе дополнительного образования с внедрением практико-ориентированного подхода, учесть возрастные особенности.

Объект исследования: обучение программированию в системе дополнительного образования.

Предмет исследования: процесс обучения школьников программированию в системе дополнительного образования.

Гипотеза исследования: если при обучении программированию на мультипарадигмальном языке программирования Python в системе

дополнительного образования применить практико-ориентированный подход, то произойдет повышение уровня мотивации обучающихся к изучению программирования.

Задачи исследования:

1. Рассмотреть суть понятия мотивации к обучению и выявить ее значимость для повышения качества образования.
2. Проанализировать современные подходы к повышению мотивации школьников в изучении информатики, особенно в аспекте обучения программированию в системе дополнительного образования.
3. Обосновать использование практико-ориентированного подхода и индивидуализации обучения для развития мотивации в курсе информатики.
4. Разработать содержание обучающего курса обучения мультипарадигмальному языку программирования Python в системе дополнительного образования для развития мотивации школьников.
5. Реализовать внедрение разработанного курса обучения мультипарадигмальному языку программирования Python в системе дополнительного образования.
6. Организовать и провести педагогический эксперимент.
7. Провести апробацию разработанной методики, целью которой является увеличение мотивации школьников к изучению программирования.

Для решения поставленных задач в данном исследовании были применены различные **методы исследования**:

– *теоретические:* понятийно-терминологический анализ; анализ и изучение научно-педагогической литературы; систематизация, обобщение;

– *эмпирические:* педагогическое наблюдение, статистическая обработка данных, анализ и обработка результатов педагогического эксперимента.

Новизна исследования состоит в разработке методики обучения языку программирования Python в системе дополнительного образования с использованием практико-ориентированности и индивидуализации обучения.

Теоретическая значимость выполненного исследования заключается в аналитическом исследовании проблемы повышения мотивации к обучению программированию с использованием практико-ориентированности и индивидуализации обучения.

Практическая значимость исследования заключается в разработке содержания курса обучения мультипарадигмальному языку программирования Python в системе дополнительного образования для развития мотивации школьников.

Структура квалификационной работы соответствует логике исследования и состоит из введения, трех глав, выводов по каждой главе, заключения, списка использованных источников и приложения.

ГЛАВА 1. ТЕОРЕТИЧЕСКИЕ АСПЕКТЫ ИСПОЛЬЗОВАНИЯ ПРАКТИКО-ОРИЕТИРОВАННОГО ПОДХОДА ДЛЯ ПОВЫШЕНИЯ МОТИВАЦИИ К ОБУЧЕНИЮ

1.1 Современные подходы к повышению мотивации в обучении

Современное общество все больше зависит от информационных технологий, которые постоянно развиваются и совершенствуются. Это делает изучение основ информатики и программирования крайне актуальным для школьников. По данным всероссийских исследований, ИТ-навыки учителей информатики, а также учеников позволяют сделать вывод о том, что информатика и программирование могут считаться одной из наиболее трудных тем, как для обучения, так и для усвоения материала.

При этом данное утверждение является справедливым как для педагогов, так и для, непосредственно, школьников. Безусловно, подобная ситуация вызвана целым комплексом факторов. Среди наиболее ключевых из них следует упомянуть, например, недостаточный опыт, которым обладают учителя информатики в сфере использования актуальных средств программирования на практике и в деятельности, связанной с образованием.

Подобная сложившаяся ситуация в сфере обучения информатике школьников, очевидно, требует определенной коррекции. Это приобретает повышенную актуальность в условиях повышения востребованности навыков программирования в повседневной жизни многих людей.

Действительно, сегодня очень сложно представить себе функциональность многих обычных предметов быта без базовых знаний основ информатики и программирования. Такими устройствами, например, являются: телевизор, смартфон, умный дом, стиральная машина. Благодаря совместному использованию умного дома с Алисой появляется возможность объединения большого количества различных устройств и

управления ими. В основе техники, используемой людьми, зачастую лежат базовые принципы информатики и программирования.

Следовательно, полноценное эффективное использование актуальной техники на сегодняшний момент практически невозможно представить без основополагающих навыков программирования, получаемых учащимися, в основной мере, на уроках информатики в основной школе [9, 10].

Большая часть учащихся склонна к освоению исключительно пользовательского курса информатики. При этом количество желающих изучать программирование традиционно не очень высоко. Причинами такого явления можно считать совокупность сложностей, вызывающих уменьшение уровня мотивации к изучению программирования. Среди них следует отметить наиболее важные:

– достаточно низкое развитие алгоритмического мышления;

– сложность понимания у большинства учащихся практических аспектов использования навыков программирования в повседневной жизни;

– низкий уровень владения иностранными языками;

– несовершенство математической подготовки у учащихся.

Безусловно, проблема, связанная с мотивацией, обладает всегда высокой актуальностью, и, в этой связи, представляет интерес для педагогов, психологов, философов. Для дальнейшего полноценного рассмотрения вопросов, связанных с формированием мотивации у школьников в ходе изучения программирования, целесообразным видится, прежде всего, предварительное раскрытие содержания самого термина «мотивация».

Согласно педагогическому словарю Г.М. Коджаспировой, под термином «мотивация» следует понимать совокупность побуждений, устойчивых мотивов, оказывающих решающее влияние на направление, содержание и характер деятельности и поведения человека. Схожее

толкование термина можно найти, например, в психологическом словаре Л.А. Карпенко, М.Г. Ярошевского и А.В. Петровского. В словаре понятие «мотивация» используется как побуждение, которое считается основной причиной активности организма и определяющей ее основную направленность.

Различные научные исследовательские работы зарубежных и отечественных психологов (например, Л.И. Божович, Л.С. Выготского, А. Маслоу, А.Н. Леонтьева, Дж. Брунер) уделяют пристальное внимание вопросам, связанным с онтогенезом мотивации, ее различным видам и функциям. Кроме этого, в них содержатся ключевые подходы к ее исследованию и созданные, опираясь на эти подходы, психологические теории учебной мотивации.

Дальнейшая конкретизация и углубление проблемы феномена «мотивации» осуществляется учеными с помощью исследования уровня становления учебной мотивации, а также ее компонентного состава. Под этим понимается совокупность условий, факторов, оказывающих преимущественное влияние на формирование мотивации как одного из важных компонентов учебной деятельности.

Основными причинами, ведущими к снижению учебной мотивации, являются:

– сложившиеся взаимоотношения в системе «ученик – учитель – родитель»;

– гормональные изменения в организме и связанные с ними изменения психического состояния школьников;

– неоднородный характер умственного развития обучающихся в школах;

– наличие заинтересованности в изучении отдельных предметов, входящих в школьный курс, а также, в некоторых случаях, отсутствие таковой;

– недостаточное понимание фундаментальных целей получения традиционного школьного образования (такая проблема характерна как для самих учеников, так и для их родителей);

– «различная» результативность учебной деятельности (одной из важных причин является, например, увеличение количества учеников в классах, что значительно усложняет возможность применения индивидуального подхода).

В соответствии с требованиями ФГОС ООО учащиеся, изучающие курс «Информатика», должны уверенно владеть навыками создания и выполнения программ для решения не очень сложных алгоритмических задач в выбранной ими среде программирования. В данном случае под программированием понимается процесс формирования упорядоченной последовательности действий для электронной вычислительной машины (компьютера).

Вся перечисленная проблематика обучения программированию широко освещена в научных работах Д.М. Гребневой, М.В, Швецкого, Д.А. Слинкина, С.А, Бешенкова и других исследователей. В тоже время современные реалии показывают, что исследований, посвященных мотивационным вопросам в обучении школьников программированию, по-прежнему не хватает.

По мнению Д.А. Слинкина метод проектов является достаточно эффективным в формировании положительной мотивации при практическом изучении программирования. В этом методе перед студентами ставится определенная проблемная область (относящаяся, обычно, к реальным жизненным ситуациям), для которой требуется формирование решения с помощью использования приобретенных навыков программирования.

В данном случае могут быть задействованы различные виды программирования: визуальное, структурное, объектно-ориентированное и другие.

Многие исследователи, например, В.К. Маркелов, О.А. Завьялова [16], Н.К. Соколов, А.Д. Емельянов [23] отмечают особую роль введения игровых элементов как усиления мотивационного эффекта.

При этом статистические данные не всегда говорят о повышении мотивации к программированию с использованием только игровых методик. Как вариант Н.А. Вальков в своей работе «Разработка компьютерных игр как мотивация к изучению алгоритмизации и программирования в основной школе» предлагает применение не напрямую игровых методик, а участие в разработке интересного контента в виде компьютерных игр [6].

Н.А. Вальков отмечает также низкую мотивацию школьников в обучении, несмотря на высокий интерес к самим компьютерным играм. В основной школе многие подростки считают, что преподаватель не имеет достаточных знаний для их обучения, а, следовательно, без должного авторитета преподавателя мотивация находится на достаточно низком уровне. Предлагается использование интересных и достаточно сложных задач, которые, однако, должны быть под силу обучающимся.

Разработка именно компьютерных игр, как считает Н.А. Вальков, сближает интересы обучающихся с необходимыми аспектами образовательной программы.

А.А. Мункоев в своей работе, связанной с изучением мотивации в обучении среди старших школьников также указывает, на изменение уровня мотивации в связи с использованием в рамках решаемых в программировании задач сюжетных задач и задач на создание компьютерных игр [18].

Особую роль в повышении уровня мотивации А.А. Мункоев отдает исследовательской деятельности, которая, на его взгляд, повышает познавательный интерес, деятельность и активности.

Проблема организации обучения информатике в разрезе программирования достаточно глубокая. Ее аспекты также отмечены в

исследованиях Е.Н. Волковой, О.М. Исаевой, А.Н. Моревой [7]. Авторы указывают, что программирование является особой технологией не в полной мере отвечающей существующим системам повышения мотивации. Так сформированы следующие заключения:

– нельзя в полной мере заменить процесс обучения программированию игровыми элементами, повторением и стандартными методиками обучения;

– обучение программированию оказывает особое влияние на развитие личности;

– мотивация в рамках обучения должна рассматриваться исключительно в аспектах получения результатов в сфере программирования, но не информатики как науки в целом;

– необходимость понимания сложности программирования для повышения внутренней мотивации обучающихся.

Кроме выявленных особенностей проблем пониженной мотивации школьников в принципе к изучению информатики в аспекте программирования важную роль также играет необходимость сдачи специальных экзаменов, содержание которых не всегда отвечает не только интересам обучающихся, но и учителей [3].

В этом смысле необходимо вернуться к особенностям социального программирования и изменениям, которое оно влечет в уровне мотивации школьников, сдающих ЕГЭ и ОГЭ.

Реальное изменение интереса к учебной деятельности возможно только с поддержкой общества и государства. Общественная позиция по отношению к данному виду деятельности должна выработать некоторый нарратив. В самом деле, нарратив становится и своеобразным методологическим плацдармом для обеспечения мотивации и интереса, и возможностью фиксации этих явлений в обществе по итогам человеческого действия [25].

Как уже говорилось выше, одной из важнейших задач информатики

является развитие познавательной деятельности учащихся. В основе которой лежит творческая активность, формирование общего (не дискретного) мировоззрения, системной и содержательной картины мира, ключевых психических качеств, коммуникативных и учебных способностей обучающихся. Причинами повышенной важности этой задачи может считаться необходимость постоянного поиска решения общеобразовательной задачи, которая состоит в увеличении качества и эффективности процесса обучения [24].

Рассмотрение эффективности в качестве одного из ключевых параметров оценки осуществляется в контексте затраченного труда и, соответственно, полученного результата. При уменьшении объема затраченных ресурсов на достижение результата и увеличении достигнутого качества целесообразно утверждать об увеличении эффективности процесса обучения, формирования, закрепления развития и иных педагогических и учебных категорий [8].

Например, при попытке мотивировать детей младшего школьного возраста только на словах к изучению практической информатики, связанной с программированием, то есть, фактически, заставлять его долгими часами изучать сложные программы, алгоритмы, коды, логику, семантику, то вряд ли будет достигнут позитивный настрой у учащегося к занятию таким актуальным, нужным, но весьма сложным делом.

Однако, если постепенно осуществлять построение с ним программных блоков на языках виртуального объектного программирования, то интерес ученика заметно повысится не только в выполнении порученных программ, но и в разработке собственного программного продукта [11].

Следует отметить, что увлеченность ученика активной познавательной деятельностью не связана с каким-либо постоянным генетическим признаком, зависящим от личности. Поэтому в данном вопросе необходимо говорить о формировании и развитии этой формы

личностной реализации.

Чаще всего, именно предметная деятельность, то есть занятия по предмету является основой для формирования интереса к восприятию окружающего мира, воспитания положительного отношения к своему труду и в целом формирования интереса к тому, что окружает ученика. Степень сознательности, прилежания, заинтересованности к работе школьника на уроке оказывает важнейшее влияние на характер их мыслей, рассуждений, действий в дальнейшем, способность к доказательству, творческому образу мышления, применению изученного в различных ситуациях [14].

Существует достаточно типичный пример. Он заключается в решении степенных уравнений инструментами любого табличного процессора. На начальном этапе может быть использовано квадратное уравнение. В ячейки рабочей таблицы осуществляется ввод только коэффициентов уравнения.

В соответствии с коэффициентами производится вычисление значения дискриминанта. В отдельной, заранее оговоренной ячейке производится вычисление корней согласно соответствующим формулам или выдается сообщение о том, что в уравнении нет действительных корней.

Далее, на втором листе выполняется работа, связанная с вычислением корней этого же уравнения, однако с помощью деления определенного выбранного отрезка на конечное количество равных частей. В данном случае задается изменение аргумента с заданным приращением с последующим вычислением значения функции, которая получена из правой части квадратного уравнения.

Затем обучаемый самостоятельно проверяет, на каком отрезке наблюдается изменение знака функции. Полученный отрезок снова делится на определенное (заранее заданное) количество частей и заново выполнить описанный выше алгоритм.

На первый взгляд вполне уместным видится вопрос о целесообразности выполнения подобной сложной многоступенчатой процедуры при существовании вполне готовой формулы, позволяющей достаточно быстро получить решение. Кроме того, сама работа приближения корней к их точным значениям – довольно трудоемкая задача.

На этом этапе учитель ставит задачу решить уравнения третьего или четвертого порядка, для которых у учеников нет готовых алгоритмов решения по формулам. Однако у учеников есть алгоритмы решения квадратных уравнений, и вполне возможно распространить эту практику на уравнения более высокого порядка.

Поощряя соревновательный метод получения результатов в самостоятельном режиме, у учителя появляется возможность наблюдения за тем, насколько быстро будут получены первые приближенные решения. Таким образом, можно с полной уверенностью утверждать об очевидном проявлении у учеников творческой активности посредством проявления устойчивого интереса.

Вообще говоря, наличие у обучающихся познавательной активности дает возможность обеспечения творческой и познавательной деятельности. В ходе этой деятельности осуществляется овладение основным содержанием учебного предмета, умениями, навыками, способами деятельности, являющимися необходимыми.

В условиях актуального (современного) образовательного пространства, когда процесс информатизации образования считается практически завершенным, целесообразно утверждать о необходимости большей степени гуманизации и гуманитарности процесса образования.

В соответствии с убежденностью многих исследователей благодаря именно такому типу реализации образовательного процесса в информационно насыщенном пространстве наибольшим образом

активизируются целевые установки учащихся на получение качественного образования.

При этом должны быть задействованы все оперативные, доступные средства, возможности которых позволяют осуществлять быстрый поиск информации не только информационного, познавательного характера, но и способен вырабатывать решение в затруднительных ситуациях (к таким можно, например, отнести решение задач по физике, математике, биологии, химии, астрономии, информатике).

Нет сомнения, что этот подход ведет к определенному стимулированию первоначального интереса. Вместе с тем, впоследствии весь имеющийся потенциал любознательной и поисковой деятельности обучающегося, как это ни парадоксально, полностью утрачивается (причем, достаточно быстро).

Если уже есть проверенное решение в готовом виде, зачем думать, искать решение, переписывать алгоритм и пошагово выполнять его, чтобы убедиться в точности полученного решения? Достаточно найти в интернете такую же или похожую задачу, переписать решение и представить его как свое собственное.

В данном случае на первый план выступает закон силы личности, который может представлять собой как качество наследственности, но в то же время означает, что творчество ребенка не подавляется, в его семье, а, наоборот, поощряется и стимулируется, когда не используется наказание при отсутствии у него на данный момент готового решения, когда у него отсутствует боязнь попросить о помощи учителя или одноклассников.

Подобный навык для ребенка является очень полезным, поскольку позволяет приучить его к коллективному взаимодействию, дающему возможность нахождения решения достаточно оперативно и рационально.

В данном случае целесообразен поиск немного другого подхода. Например, предлагать обучающемуся только те задачи, для которых у них вообще нет готовых решений. Это могут быть, например, задачи,

связанные с построением, рисованием, изготовлением проекта, а также так называемые кейс-задачи.

Безусловно, подобный вид организации образовательной деятельности видится достаточно затруднительным для самого учителя. Он, в данном случае, сталкивается с необходимостью достижения оригинальности, придумывая и предлагая задачи, которые, очевидно, не являются общедоступными.

В зависимости от возраста учеников источником для таких заданий могут быть классические учебники или сборники задач с последующим небольшим изменением взятых оттуда задач (изменение числовых значений, перефразировка смысловой гаммы). Необходимо отметить, что с повышением возраста обучаемых возрастает и уровень отрыва формулировки предлагаемого задания от классической формулировки.

Практически отсутствуют готовые решения для таких классических задач какими представляет собой комплекс задач по информатике, относящихся к освоению программного обеспечения или определенных функций программного продукта в самостоятельном режиме. Еще большей привлекательностью обладает вопрос, связанный с формированием задания или его отдельного фрагмента, в котором использован (в прямой или косвенной форме) материал изученный ранее.

В одном из подобных вариантов может быть предложена задача по физике, включающая несколько вопросов. Например, это может быть такая задача: определение выталкивающей силы на заданной глубине; определение массы объекта на заданной глубине с учетом выталкивающей силы; размещение максимально возможного груза для достижения разрыва; расчет силы натяжения троса или веревки, на который подвешен груз; определение запаса прочности, рассчитываемого в процентах.

После этого с помощью определенной практической ситуации осуществляется приведение всех вышеописанных задач к одной. Например, если необходимо опустить глубоководный навигационный

корпус заданной массы на заданную глубину, используются стальные тросы, свойства которых (плотность стали, площадь поперечного сечения и разрывная нагрузка) известны. Для расчета толщины тросов на борту необходимо создать модель, учитывающую запас прочности в 60%.

Для решения этой задачи можно использовать ранее приведенные решения, алгоритмы и формулы. Вместе с преподавателем обучающиеся имеют возможность создать алгоритм программирования этой задачи, используя один из структурных (визуальных, объектно-ориентированных) языков программирования или электронную таблицу.

Во всех вышеперечисленных случаях необходимо понимать, что существует определенный порог готовности обучающегося к преодолению трудностей. В случае постоянного столкновения ученика с неразрешимыми (хотя бы в его понимании) задачами, вместо развития интереса к образовательному процессу вполне вероятно наступление регресса. Безусловно, вопросы формирования деятельностной и познавательной активности должны впервые подниматься еще в дошкольном возрасте. Если этого не сделано, то рассуждать о развитии второй ступени обучения в некоторых ситуациях может быть бессмысленно.

К сожалению, подобные ученики встречаются в школах и достаточно часто. Чтобы достичь приемлемого качества обучения со стороны ответственного, творческого педагога требуется раскачивать подобных обучающихся, практически принуждать их, пытаться находить привлекательность, относящуюся к сфере личной заинтересованности ребенка.

Данный процесс обладает достаточно высокой трудоемкостью, и далеко не каждый из педагогов способен грамотно его осуществлять с первых лет работы. Поэтому невозможно переоценить опыт, источником которого выступает взаимодействие с педагогами со стажем и методистами.

Кроме того, учитывая одну из задач информатизации – выявление и развитие творческого потенциала нового поколения – весьма целесообразно распространять этот опыт во времени, в деятельности учителей различных поколений и различных предметных и научных направлений.

1.2 Проблема обучения программированию в системе дополнительного образования

Существует достаточно много методов, направленных на активизацию познавательной деятельности учащихся на уроках информатики, связанных с основами алгоритмизации и программирования. Представляется целесообразным рассмотреть некоторые из них более подробно.

Первый метод ориентирован на обращение к жизненному опыту учащихся. Суть данного метода в следующем: учителем осуществляется обсуждение с учащимися достаточно хорошо знакомых им ситуаций. Для полного восприятия их требуется изучение предлагаемого материала. При этом необходимым является обеспечение действительной «жизненности» предлагаемых ситуаций (любая надуманность должна быть, по возможности, исключена).

В действительности обращение к жизненному опыту должно рассматриваться не только в качестве приема для формирования мотивации. Более важным представляется, что учащиеся могут увидеть применимость получаемых ими знаний в практической деятельности. Действительно, одной из значимых проблем при изучении многих дисциплин является то обстоятельство, что учащиеся не могут вообще понять, каким образом ими могут применяться получаемые знания на практике [2].

Второй метод состоит в создании проблемной ситуации, либо

разрешении парадоксов. Многие считают данный метод, в значительной степени, универсальным. Учителем осуществляется постановка перед учащимися определенной проблемы, в ходе преодоления которой и происходит осваивание учащимся тех знаний, навыков и умений, которые представляются необходимыми для усвоения в соответствии с программой.

В качестве примера можно привести урок по изучению табличного процессора Excel. Учитель дает задание на изучение функций данного программного продукта. Ученикам необходимо написать, посредством каких функций осуществляется то или иное действие. Например, для форматирования текста в отдельной ячейке можно задействовать шрифт, размер, расположение текста, формат числа (при необходимости). После окончания исследования учениками табличного процессора учитель выслушивает все ответы с последующим указанием на ряд функций, о которых в ответах учеников не было ничего упомянуто. Таким образом и создается проблемная ситуация [15].

Нельзя кратко не сказать о таком методе, как ролевой подход. Группе учащихся (или одному учащемуся) предлагается сыграть роль определенного действующего лица, например, формального исполнителя алгоритма. В соответствии с заданной ролью происходит конкретизация внимания именно на тех существенных условиях, усвоение которых и представляется в качестве учебной цели. Если, например, рассматривается такая конструкция, как «цикл», то речь идет о точном, неукоснительном выполнении команд, выполняющих реализацию данной конструкции.

В качестве примера можно привести урок с учащимися 8-9-х классов в виде ролевой игры на тему «Судебный процесс над информационными технологиями». Соответственно, реализация данной игры предполагает выбор из группы учащихся «обвинителей», «членов суда», «защитников» и, непосредственно, председателя судебного процесса. Оставшиеся учащиеся играют роль «зрителей» в зале суда. Ключевая идея в этой игре

заключается в обязательности рассмотрения всех сторон – и хороших, и плохих в информационных технологиях. Подобный подход даст учащимся возможность определения для себя позитивных и негативных аспектов информационных технологий.

Также важным методом является, несомненно, решение задач нестандартного характера на логику и смекалку. Подобные задачи предлагаются учащимся. Они рассматриваются либо как разминка в начале урока, либо разрядка, смена вида работы в течение урока, иногда – дополнительное задание в рамках домашней работы.

Например, можно упомянуть о таком алгоритме шифрования слов, как метод Цезаря. В соответствии с ним каждая буква текста заменяется на другую. Для замены определенной буквы требуется сместиться от исходной буквы на заранее оговоренное число символов. Следует, однако, учитывать, что чтение алфавита осуществляется по кругу, и, если при счете достигнут конец алфавита, то счет продолжается с первой буквы алфавита «А».

В качестве задания ученикам предлагается, например, расшифровать слово «МТКСФРЕТВЦКБ», кодирование которого выполнено шрифтом Цезаря. Является известным, что каждую букву в исходном тексте заменили на вторую, после нее, буквой. Несложно увидеть, что правильным ответом будет слово «Криптография», то есть наука, изучающая принципы, средства и методы преобразования информации для обеспечения ее защиты от искажения и несанкционированного доступа.

Важным элементом освоения любого учебного материала, безусловно, является игра. Деловая игра, основанная на ролевом подходе, является эффективным методом обучения, который позволяет ученикам не только получить новые знания, но и применить их на практике. В рамках такого урока каждый учащийся примеряет на себя определенную роль, что позволяет им глубже понять и освоить материал.

Подготовка к деловой игре является важным и неотъемлемым этапом

ее проведения. Учителю необходимо тщательно продумать сценарий игры, определить цели и задачи, выделить роли для каждого ученика и разработать материалы, необходимые для ее проведения. Также важно сформировать ясные правила и структуру игры, чтобы все учащиеся могли понять и выполнять свои роли, благодаря этому может гарантироваться успех подобной формы занятий у учащихся.

Перед началом игры весь класс разделяется на определенное количество команд. Соревнование команд видится наиболее эффективным методом проведения игры.

Например, заявлена игра «Я – будущий программист». Для такой игры целесообразно разделить класс на две команды. В игре предполагается проведение трех туров. В первом туре проводится конкурс «Викторина», в которой учитель задает вопросы определенной категории, на который ученики дают ответы и, соответственно, набирают баллы. Ответы на вопросы даются командами по очереди. В случае отсутствия правильного ответа у одной из команд, осуществляется переход права на ответ к другой команде.

Следующий, второй тур, представляет собой соревнование между капитанами команд. Они по очереди выходят к доске и делают попытку сбора алгоритма какой-либо представленной ситуации или действия. Например, предлагается разработка алгоритма наливания кофе. Наконец, третий, последний тур может являться творческим заданием. После этого производится подсчет итогов. При этом команда, у которой набранных баллов больше, и объявляется победителем.

Следовательно, проблематика формирования мотивации учащихся в ходе обучения программированию, безусловно, обладает сложностью, актуальностью и неоднозначностью. Результаты изучения педагогической практики свидетельствуют, с одной стороны, о наличии отдельных методов, обладающих большей эффективностью в повышении мотивации у обучающихся при изучении программирования [20].

С другой стороны, существует ряд сложностей при обучении школьников информатике, например, выделение недостаточного количества часов на изучение информатики в школе, малое количество материалов дидактического характера. С учетом несомненной актуальности указанных проблем в качестве одного из выходов видится изучение теста в контексте темы и применения педагогических технологий в мероприятиях внеклассного характера.

Вообще говоря, следует помнить, что потребность в получении новых знаний закладывается в детях самой природой. Вместе с тем, наблюдается тенденция снижения этой потребности с возрастом в случае «переполнения» ребенка информацией.

Для повышения заинтересованности школьников целесообразно использование их потребностей, являющихся характерными для определенного возраста: потребность в общении, в самореализации, в самовыражении; потребность в принципиально новых видах деятельности. Именно эти группы потребностей часто являются определяющими в повышении мотивации учебной деятельности.

Кроме предложенных выше подходов, существует ряд других способов, позволяющих эффективно повысить мотивацию учеников:

– обеспечение у учащихся переживания успеха в деятельности. Для этого требуется корректный подбор уровня сложности заданий и максимально объективная оценка результата;

– организация сотрудничества с учениками на уроке, а также взаимопомощи и позитивного отношения к предмету в целом;

– это индивидуальный подход к каждому учащемуся. Учителям необходимо учитывать индивидуальные особенности и потребности каждого ученика, адаптировать содержание образовательной программы и методики обучения. Это позволит каждому ученику ощутить свою

значимость и почувствовать поддержку на пути к достижению своих учебных целей;

– корректность формирования учителем отношений с учениками, заинтересованность в их успехах;

– использование всех возможностей учебного материала для повышения заинтересованности учащихся. Для этого необходима постановка проблем, активизация самостоятельного мышления.

При этом формы и методы ведения урока должны быть выбраны таким образом, чтобы максимально вовлечь ученика в сам процесс обучения. В результате совместной работы учителя и ученика должна быть обеспечена интерактивность урока [5].

В ходе обучения ученик должен выполнять роль субъекта учения, то есть обладать потребностью и желанием в познании учебного материала и последующим применением его на практике. Решение данной задачи представляется невозможным без формирования соответствующей мотивации. Другими словами, в начале изучения любого нового материала должно быть именно формирование мотивации.

Как известно, в течение достаточно долгого времени в качестве ключевого мотива для изучения информатики рассматривался интерес к компьютеру. Пока это обладало неким налетом новизны и неизведанности, очевидно, существовал повышенный интерес у детей. Действительно, компьютер мог выступать в качестве помощника и друга, обладал способностью развлечь и связать с окружающим миром. Вместе с тем, большая часть детей воспринимает компьютер сейчас вполне обычно, поэтому они изначально не испытывают особого интереса к предмету «информатика».

В данном контексте целесообразно указать известную классификацию методов обучения в соответствии с уровнем активности обучающихся:

– обучающийся играет роль «объекта обучения». Перед ним стоит задача усвоения и последующего воспроизведения материала, передаваемого ему учителем (который в данном случае выступает как источник правильных знаний). Такой подход называется пассивным;

– обучающийся выступает в роли «субъекта обучения». Другими словами, ему необходимо выполнять задания творческого характера, вести активный диалог с учителем. В качестве основных методов используются: вопросы от ученика педагогу, цель которых состоит в развитии творческого мышления; творческие задания (зачастую, домашние). Соответственно, такой подход называется активным.

В данном контексте целесообразно рассмотреть еще несколько методов, которые позволяют повысить мотивацию учеников к изучению информатики и программирования.

Первый из таких методов уже упоминался – это обращение к жизненному опыту учащегося. Его теоретические основы уже были описаны выше. В качестве наглядного примера его использования можно привести, например, при изучении темы «Классификация».

Основание классификации, принцип деления учащихся в школе по классам. Действительно, данный пример хорошо дает понять, каким образом осуществляется распределение в школе учащихся по классам; что именно выступает в качестве основания при подобном распределении объектов. Еще один наглядный пример – изучение темы «База данных. СУБД».

Здесь целесообразно дать ученикам работу практического характера, заключающуюся в создании телефонной книги. Это не должно вызвать особых затруднений и непонимания, поскольку практически все ученики в школе обладают мобильными телефонами.

Еще одним, ранее не рассматриваемым методом, является проектно-исследовательская деятельность.

На сегодняшний день метод проектов достаточно хорошо исследован

и нашел свою нишу у педагогов при проведении уроков. В частности, он является вполне уместным при изучении информатики и программирования.

Не секрет, что разработка проекта представляет собой достаточно сложный и трудоемкий процесс. Вместе с тем, он, очевидно, стимулирует детей к поиску ответов на те или иные вопросы. Преимущественно, работа подобного характера заинтересовывает всех учащихся без исключений. Благодаря такому виду учебной деятельности у учеников происходит развитие логического мышления, формирование умений и навыков общеучебного характера.

В ходе демонстрации собственных проектов учащиеся приобретают определенный опыт публичных выступлений, который, без сомнения, пригодится им в дальнейшем. При вовлечении учащегося в работу творческого характера у него происходит развитие умения собирать информационно-иллюстративный материал в самостоятельном режиме, проявления своего творческого начала. Наиболее важным в этой связи видится появление удовлетворения от результатов своего труда и чувства уверенности в собственных способностях и силах.

Чтобы обеспечить на устойчиво высоком уровне мотивацию учебной деятельности, целесообразно применение проектов на каждой из ступеней обучения информатике, начиная с мини-проектов в среднем звене.

Нельзя также забывать о межпредметной связи, которая вполне проявляется в методе проектов. Например, на уроках географии при изучении климата учащиеся с помощью наблюдений заполняют таблицу. Затем, после получения знаний на уроках информатики по теме «Табличный процессор Excel» они уже способны осуществить создание диаграмм и наглядно увидеть изменения погодных условий.

Следовательно, ученики чувствуют себя полноценными исследователями, что делает актуальность в получении знаний по изучаемым темам очевидной, не упоминая уже о вполне очевидной

практической ценности подобной работы.

Выводы по главе 1

Современные информационные технологии, которые окружают школьника интересны большинству только в рамках их использования. Необходимость подготовки новых ИТ-специалистов предполагает увеличение числа абитуриентов, которые обладают достаточными знаниями в программировании для получения соответствующих специальностей. Вместе с тем, на данный момент наблюдается понижение уровня мотивации в обучении информатике, следствием которого становится ухудшение знаний и навыков в программировании.

Поиски путей повышения мотивации школьников в изучении информатики выявили некоторые особенности, связанные со спецификой программирования. Так нельзя в полной мере заменить процесс обучения программированию игровыми элементами, повторением и стандартными методиками обучения, при этом обучение программированию оказывает особое влияние на развитие личности и мотивация в рамках обучения должна рассматриваться исключительно в аспектах получения результатов в сфере программирования, но не информатики как науки в целом. Возникает вопрос о необходимости понимания сложности программирования для повышения внутренней мотивации обучающихся.

Наиболее актуальным видится использование в процессе обучения программированию практической направленности решаемых задач и формирование индивидуальных траекторий для обучающихся, которые обладают достаточными способностями в области информационных технологий, с целью поддержания их познавательного интереса и получения более высоких уровней достижений в процессе освоения программирования. Важным аспектом такой деятельности является использование средств из системы дополнительного образования.

ГЛАВА 2. МЕТОДИКА ПРИМЕНЕНИЯ ПРАКТИКО-ОРИЕНТИРОВАННОГО ПОДХОДА В ОБУЧЕНИИ ПРОГРАММИРОВАНИЮ В СИСТЕМЕ ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ

2.1 Отбор содержания для методики практико-ориентированного подхода в обучении программированию

В теоретической части работы нами было определено, что повышение мотивации обучающихся к изучению программирования в значительной степени зависит от применения образовательных подходов и методик в учебном процессе на уроках информатики. Также мы рассмотрели методы повышения мотивации у обучающихся, которые напрямую связаны с содержанием тем по информатике. В данном этапе работы необходимо провести анализ влияния планируемых результатов освоения образовательной программы на уровень мотивации обучающихся программированию. Основной целью данного анализа является выявление положительных эффектов, которые результаты образовательной программы оказывают на мотивацию студентов.

В рамках нашей исследовательской работы мы исследовали авторскую программу Л. Л. Босовой, А. Ю. Босовой за 7-9 классы, взятую нами в качестве основы для определения содержания курса информатики в аспекте обучения программирования [4].

В процессе изучения темы «Алгоритмы и программирование» предмета «информатика» в основной и старшей школе в аспекте программирования формируются личностные результаты:

– сформированность мировоззренческих представлений об информации, информационных процессах и информационных технологиях, соответствующих современному уровню развития науки и

общественной практики и составляющих базовую основу для понимания сущности научной картины мира;

– интерес к обучению и познанию, любознательность, готовность и способность к самообразованию, осознанному выбору направленности и уровня обучения в дальнейшем;

– овладение основными навыками исследовательской деятельности, установка на осмысление опыта, наблюдений, поступков и стремление совершенствовать пути достижения индивидуального и коллективного благополучия;

– сформированность информационной культуры, в том числе навыков самостоятельной работы с учебными текстами, справочной литературой, разнообразными средствами информационных технологий, а также умения самостоятельно определять цели своего обучения, ставить и формулировать для себя новые задачи в учёбе и познавательной деятельности, развивать мотивы и интересы своей познавательной деятельности.

Основными метапредметными результатами, формируемыми при изучении темы «Алгоритмы и программирование» предмета «информатика» в основной школе в аспекте программирования, являются:

– умение создавать, применять и преобразовывать знаки и символы, модели и схемы для решения учебных и познавательных задач;

– самостоятельно выбирать способ решения учебной задачи (сравнивать несколько вариантов решения, выбирать наиболее подходящий с учётом самостоятельно выделенных критериев).

– самостоятельно составлять алгоритм решения задачи (или его часть), выбирать способ решения учебной задачи с учётом имеющихся ресурсов и собственных возможностей, аргументировать предлагаемые варианты решений;

– составлять план действий (план реализации намеченного алгоритма решения), корректировать предложенный алгоритм с учётом получения новых знаний об изучаемом объекте.

Основными предметными результатами, формируемыми при изучении темы «Алгоритмы и программирование» предмета «информатика» в основной и старшей школе в аспекте программирования, являются умения:

– раскрывать смысл понятий «исполнитель», «алгоритм», «программа», понимая разницу между употреблением этих терминов в обыденной речи и в информатике;

– описывать алгоритм решения задачи различными способами, в том числе в виде блок-схемы;

– составлять, выполнять вручную и на компьютере несложные алгоритмы с использованием ветвлений и циклов для управления исполнителями, такими как Робот, Черепашка, Чертёжник;

– использовать константы и переменные различных типов (числовых, логических, символьных), а также содержащие их выражения, использовать оператор присваивания;

– использовать при разработке программ логические значения, операции и выражения с ними;

– анализировать предложенные алгоритмы, в том числе определять, какие результаты возможны при заданном множестве исходных значений;

– создавать и отлаживать программы на одном из языков программирования (Python, C++, Паскаль, Java, C#, Школьный Алгоритмический Язык), реализующие несложные алгоритмы обработки числовых данных с использованием циклов и ветвлений, в том числе реализующие проверку делимости одного целого числа на другое, проверку натурального числа на простоту, выделения цифр из натурального числа.

При отборе содержания для методики практико-ориентированного подхода в обучении программированию необходимо руководствоваться в первую очередь достижением личностных, предметных и метапредметных результатов освоения курса информатики.

2.2 Методика обучения программированию на языке Python в системе дополнительного образования

При разработке практико-ориентированных задач в методике обучения программированию следует руководствоваться следующим алгоритмом:

1. Определяем личностные, метапредметные и предметные результаты обучения по выбранным темам курса программирования.
2. Устанавливаем критерии практической направленности, которые являются наиболее важными.
3. Формулируем задания по теме курса, которые будут отвечать требованиям практико-ориентированного подхода, исходя из личностных, метапредметных и предметных результатов обучения.

Необходимо придерживаться определенной структуры представления каждого занятия:

- тема;
- теоретический материал с практическим применением в области каждого термина и понятия;
- содержательный блок, сформулированный в виде практических задач;
- практическая работа для самостоятельного выполнения.

Цель практико-ориентированного подхода: создание жизненных ситуаций для применения полученных навыков программирования.

Нами был создан учебный курс «Основы программирования на Python» на 34 учебных занятия по 1 академическому часу для обучения

программированию на мультипарадигмальном языке программирования Python с практико-ориентированным подходом (таблица 1).

Курс рассчитан на школьников, начиная с 7 класса. Однако, данной методикой можно пользоваться при обучении старших школьников, а также студентов, не знакомых ранее с программированием, так как входные требования к изучению курса в области знаний программирования не предъявлены. То есть обучение начинается «с нуля».

По итогу прохождения курса обучающиеся будут проходить контрольное тестирование в области предметных знаний в программировании на языке Python. Контрольный тест представлен в приложении 3.

При применении методики практико-ориентированного подхода к обучению программированию важно каждый запуск программного кода проводить сначала ученикам, и только затем учителю для демонстрации работы. При таком подходе будет развиваться навык проверки структуры кода на синтаксические ошибки, такие как: пропуски знаков, пропуски табуляции, вложение тела функции и отступы внутри условного ветвления. Этим обеспечивается работоспособность кода на каждом этапе решения задачи. Исключаются не только логические ошибки, но и ошибки правописания, синтаксиса и опечатки.

Каждое занятие рассчитано на один академический час. В практических занятиях обучающимся предлагается практическая работа для самостоятельного выполнения. Практическая часть выполняется с обсуждением в группе и с преподавателем. Задачи практической части носят характер бытового содержания, который можно обсудить дома и перепрограммировать на выполнение, исходя из жизненного уклада семьи. Например, задача на подбор кино для просмотра может быть выполнена исходя из любимого жанра кинофильмов каждого члена семьи. Задача на вычисление возраста будет решаться по дате рождения родственников и других членов семьи.

Некоторые задачи предлагаются, исходя из школьной программы по другим предметам: математика, физика и др. Например, для выполнения домашних заданий по математике будет написан программный код для вычисления площади геометрических фигур, который позволит делать проверку математических задач за несколько секунд. Однако, чтобы углубить навыки программирования, после детального разбора практических задач, основанных на применении их в жизненном опыте обучающихся, предлагается решение некоторых абстрактных задач, которые будут подготавливать к формулировкам задач, встречающихся на ОГЭ и ЕГЭ. Таким образом, на основе практико-ориентированного подхода будет стираться страх перед сложной формулировкой задач высокого уровня, а значит, мотивация к дальнейшему изучению программирования не будет снижаться.

Конспекты с учебным материалом представлены в Приложении 1.

Таблица 1 – Тематическое планирование курса по программированию

№ п/п	Наименование разделов	Кол-во часов	Из них	
			Теория	Практика
1	Введение	5	2	3
1.1	Знакомство с Python	2	1	1
1.2	Переменные и типы данных.	2	1	1
1.3	Ввод и вывод данных в Python	1	-	1
2.	Алгоритмизация	7	3	4
2.1	Логические выражения и операторы	2	1	1
2.2	Условный оператор и ветвление.	1	-	1
2.3	Множественное ветвление	2	1	1
2.4	Циклы в программировании	2	1	1
3.	Функции	9	4	5
3.1	Функции в программировании	2	1	1
3.2	Локальные и глобальные переменные	2	1	1
3.3	Возврат значений из функции. Оператор return	2	1	1
3.4	Параметры и аргументы функции	1	-	1
3.5	Встроенные функции	2	1	1
4.	Модули	3	1	2
4.1	Модули	2	1	1
4.2	Генератор псевдослучайных чисел	1	-	1

Продолжение таблицы 1

5.	Работа с классами данных	10	4	6
5.1	Списки	2	1	1
5.2	Цикл for	1	-	1
5.3	Строки	1	-	1
5.4	Кортежи	2	1	1
5.5	Словари	2	1	1
5.6	Файлы	2	1	1
	Итого часов:	**34**	**14**	**20**

Система заданий разработана с нарастающей сложностью: от самых простых понятий и типов переменных, переходя к работе с циклами, списками, словарями. Что даёт прочную основу для дальнейшего изучения программирования.

Примеры практико-ориентированных задач:

Тема «Ветвление. Условный оператор».

Проверить хватит ли покупателю 99 рублей на покупку двух товаров. Пример кода с веткой else на языке программирования Python:

```
tovar1 = 50
tovar2 = 32
if tovar1+ tovar2 > 99 :
        print("99 рублей недостаточно")
else:
   print("Чек оплачен")
```

Тема «Множественное ветвление: операторы if-elif-else.»

Необходимо написать программу, которая рекомендует пользователю определенный видео-контент, исходя из возраста пользователя. Возраст пользователя будет попадать в одну из четырех возрастных групп, например: от 3 до 6 лет, от 6 до 12, от 12 до 16, 16+. Соответственно, сначала необходимо запросить возраст пользователя. А далее решить поставленную задачу, используя только конструкцию if-else?

```
age = int(input('Сколько Вам лет? '))
```

```
print('Предлагаем к просмотру: ')
if 3 <= age < 6:
   print('"Смешарики"')
if 6 <= age < 12:
   print('"Трансформеры"')
if 12 <= age < 16:
   print('"Я - робот"')
if 16 <= age:
print('"Области тьмы"')
```

Тема «Циклы в программировании. Цикл while».

Какая будет сдача у покупателя после покупки каждого следующего товара, пока не закончатся 100 рублей?

```
total = 100
while total > 0:
   n = int(input("Введите стоимость покупки"))
   total = total - n
print("Ресурс исчерпан")
```

Тема «Функции в программировании».

Директору зоопарка необходимо посчитать еду для животных. При этом, расчет будет идти один и тот же, но животные указаны разные. В результате мы получим огромный код, в котором будет постоянно повторяться одинаковый участок кода.

```
print("Сколько сена и листьев для жирафов?")
a = int(input("Введите значение: "))
b = int(input("Введите значение: "))
print("Итого", a+b)
print("Сколько мяса и овощей для гепардов?")
a = int(input("Введите значение: "))
b = int(input("Введите значение: "))
print("Итого", a+b)
```

```
print("Сколько малины и мёда для медведей?")
a = int(input("Введите значение: "))
b = int(input("Введите значение: "))
print("Итого", a+b)
```

Для того, чтобы решить проблему множественного дублирования кода в разных местах программы, будем использовать функции. С их помощью, мы будем выполнять один и тот же расчет тогда, когда это требуется в задаче.

Тема «Локальные и глобальные переменные».

Если в какой-то задаче программе выполняет промежуточные вычисления, которые в общем-то пользователю не нужно выводить на экран, то такие вычисления удобно где-то сохранять, так как дальнейшая работа программы построена на этих промежуточных значениях. В таких случаях удобно использовать глобальные переменные. Построим следующую программу:

```
itog = 0
def prymougolnik():
    a = float(input("Укажите ширину: "))
    b = float(input("Укажите высоту: "))
    itog = a*b
def treugolnik():
    a = float(input("Укажите основание: "))
    h = float(input("Укажите высоту: "))
    itog = 0.5 * a * h
vibor = input("Выберите фигуру: 1-прямоугольник, 2-треугольник: ")
if vibor == '1':
        prymougolnik ()
elif vibor == '2':
        treugolnik ()
print("Площадь фигуры: %.2f" % itog)
```

В нашей программе мы создали глобальную переменную itog, которой присвоили значение 0. В теле функций в эту переменную помещается результат вычислений. А в конце программы значение нашей глобальной переменной будет выведено на экран. Давайте посмотрим, как будет работать наш код.

Выберите фигуру: 1-прямоугольник, 2-треугольник: 2

Укажите основание: 6

Укажите высоту: 4.5

Площадь фигуры: 0.00

Давайте выясним, почему результат всё еще 0.

Выводы по главе 2

Во второй главе исследования рассматриваются личностные, метапредметные и предметные результаты, которые должны быть достигнуты обучающимися на уроках информатики в основной школе при изучении тем по алгоритмизации и программированию.

На основе этих результатов определяется практико-ориентированное наполнение задач по курсу обучения программирования

Впоследствии разработанные нами задачи были использованы в процессе обучения программированию учащихся АНО «Центр информационных технологий «РазвИТие» в целях повышения мотивации к обучению программирования.

ГЛАВА 3. ОРГАНИЗАЦИЯ И ПРОВЕДЕНИЕ ПЕДАГОГИЧЕСКОГО ЭКСПЕРИМЕНТА ДЛЯ ОЦЕНКИ ЭФФЕКТИВНОСТИ ПРИМЕНЕНИЯ МЕТОДИКИ ОБУЧЕНИЯ ЯЗЫКУ ПРОГРАММИРОВАНИЯ PYTHON В СИСТЕМЕ ДОПОЛНИТЕЛЬНОГО ОБРАЗОВАНИЯ

3.1 Организация и проведение педагогического эксперимента

В первом разделе исследования мы провели тщательный анализ литературы, посвященной рассматриваемой проблеме повышения мотивации обучающихся к изучению программирования в системе дополнительного образования.

В практической части исследование предполагало:

– проведение анализа результатов обучающихся при изучении программирования в курсе информатики;

– разработку содержания методики обучения языку программирования Python в системе дополнительного образования;

– реализацию методики обучения языку программирования Python в системе дополнительного образования;

– разработку анкеты для определения уровня мотивации школьников к изучению программирования.

Педагогический эксперимент проводился на площадке АНО «Центр информационных технологий «РазвИТие» города Снежинска.

Цель экспериментальной работы: доказать, что если при обучении программированию на мультипарадигмальном языке программирования Python в системе дополнительного образования применить практико-ориентированный подход, то произойдет повышение уровня мотивации обучающихся к изучению программирования.

Для достижения поставленной цели были сформулированы следующие задачи:

1. Выявить текущий уровень мотивации школьников к изучению программирования с помощью разработанной анкеты-опросника.

2. Разработать методику обучения языку программирования Python в системе дополнительного образования с практико-ориентированным подходом.

3. Экспериментально проверить и доказать правдоподобность поставленной гипотезы, с помощью методов математической статистики.

Этапы исследования. При проведении экспериментальной работы выделяли три этапа в течение 2021-2024 гг.

В рамках первого этапа, протекающего с 2021 по 2022 годы, был проведен анализ нормативно-правовой документации и психолого-педагогической литературы по вопросам повышения мотивации школьников к изучению программирования. Обосновывалось использование методики практико-ориентированного подхода к изучению программирования в системе дополнительного образования.

На втором этапе (2022-2023 гг.) проведено глубокое исследование содержание школьного курса информатики в обучении программированию; разрабатывалась методика обучения языку программирования Python в системе дополнительного образования.

На третьем этапе (2023-2024 гг.) определялся уровень мотивации к изучению программирования учащихся АНО «Центр информационных технологий «РазвИТие» на констатирующем этапе эксперимента. Была осуществлена экспериментальная проверка гипотезы, направленной на проверку ее правдоподобности. Для достижения данной цели были использованы методы математической статистики, которые позволили собрать и проанализировать необходимые данные, В результате проведенных исследований были получены важные выводы, которые четко сформулированы и оформлено диссертационное исследование.

3.2 Анализ результатов применения методики обучения языку программирования Python в системе дополнительного образования

Для того, чтобы проанализировать текущий уровень мотивации к изучению программирования учеников, нами был осуществлен опрос учащихся АНО «Центр информационных технологий «РазвИТие». Для диагностики уровня мотивации школьников нами были сформулированы показатели мотивации. Учащимся было предложено провести самооценку уровня мотивации к изучению программирования ответив на вопросы анкеты, в которых можно было оценить каждый ответ от 1 до 5 баллов.

Анкета, которую заполняли учащиеся, состояла из 13 вопросов. Каждый вопрос направлен на оценку уровня мотивации ученика по определенным критериям. Учащимся предлагалось поставить себе от 1 до 5 баллов в ответ на каждый вопрос. Эта анкета способствует определению степени мотивации учеников и позволяет выявить их сильные и слабые стороны в этом аспекте. Каждый вопрос создан таким образом, чтобы ученик мог честно оценить свою мотивацию и взвесить, насколько его интересует данная тема или задача. Это важное средство для самооценки и поможет ученикам более осознанно принимать решения и определить свои приоритеты.

Интерпретация уровней мотивации в ответах на вопросы анкеты:

– 1 – мотивация отсутствует (очень низкий уровень мотивации);

– 2 – мотивация незначительная (низкий уровень мотивации);

– 3 – мотивация недостаточна (средний уровень мотивации);

– 4 – мотивация достаточно высокая (уровень мотивации выше среднего);

– 5 – мотивация ярко выраженная (высокий уровень мотивации).

Опросный лист анкеты с самооцениванием обучающихся представлен в Приложении 2.

В ходе проведенного анкетирования приняло участие 10 учеников.

При подсчете суммы баллов, отражающих уровень мотивации, были выявлены ученики с разным уровнем мотивации.

Для выявления уровня мотивации набранные баллы по всем вопросам суммировались.

Распределение по уровням, исходя из набранных баллов:

– от 13 до 19 – мотивация отсутствует (очень низкий уровень мотивации);

– от 20 до 31 – мотивация незначительная (низкий уровень мотивации);

– от 32 до 45 – мотивация недостаточна (средний уровень мотивации);

– от 46 до 58 – мотивация достаточно высокая (уровень мотивации выше среднего);

– от 59 до 65 – мотивация ярко выраженная (высокий уровень мотивации).

При проведении анкетирования обучающихся на констатирующем этапе эксперимента мы получили результаты, которые отражены в таблице (таблица 2).

Таблица 2 – Выявление уровня мотивации учащихся к изучению программирования на констатирующем этапе эксперимента

№ п/п	Обучающийся	Сумма баллов	Уровень мотивации
1	Обучающийся 1	25	мотивация незначительная (низкий уровень мотивации)
2	Обучающийся 2	30	мотивация незначительная (низкий уровень мотивации)
3	Обучающийся 3	23	мотивация незначительная (низкий уровень мотивации)
4	Обучающийся 4	35	мотивация незначительная (низкий уровень мотивации)
5	Обучающийся 5	28	мотивация незначительная (низкий уровень мотивации)
6	Обучающийся 6	27	мотивация незначительная (низкий уровень мотивации)
7	Обучающийся 7	21	мотивация незначительная (низкий уровень мотивации)

Продолжение таблицы 2

8	Обучающийся 8	29	мотивация незначительная (низкий уровень мотивации)
9	Обучающийся 9	32	мотивация незначительная (низкий уровень мотивации)
10	Обучающийся 10	24	мотивация незначительная (низкий уровень мотивации)

На констатирующем этапе эксперимента выявленный уровень мотивации учащихся представлен в виде диаграммы на рисунке 1.

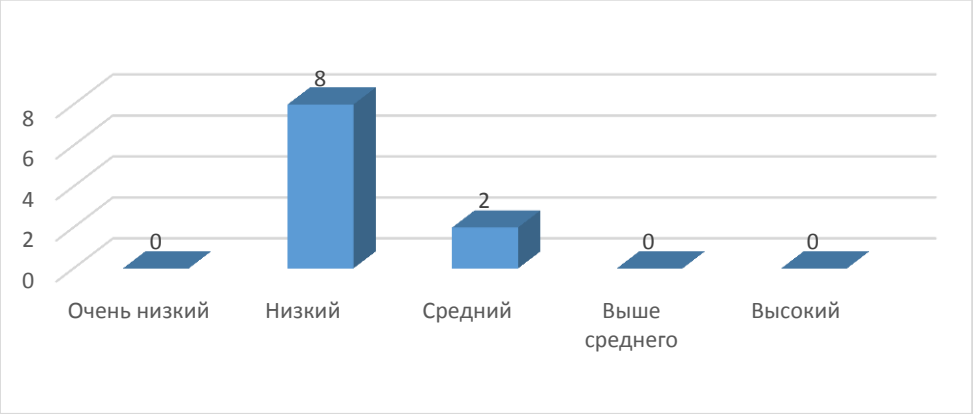


Рисунок 1 – Мотивация учащихся на констатирующем этапе эксперимента

По результатам анкетирования видно, что у учащихся уровень мотивации в основном находится на низком уровне. У трёх учащихся уровень мотивации находится на очень низком уровне.

На формирующем этапе эксперимента в учебный процесс АНО «Центр информационных технологий «РазвИТие» был внедрен разработанный учебный курс «Основы программирования на Python».

После завершения курса было проведено повторное анкетирование с целью проверки уровня мотивации учащихся.

Результаты анкетирования на контрольном этапе эксперимента представлены в таблице (таблица 3).

Таблица 3 – Измерение уровня мотивации учащихся на контрольном этапе эксперимента

№ п/п	Обучающийся	Сумма баллов	Уровень мотивации
1	Обучающийся 1	45	мотивация недостаточна (средний уровень мотивации)
2	Обучающийся 2	50	мотивация достаточно высокая (уровень мотивации выше среднего)
3	Обучающийся 3	41	мотивация недостаточна (средний уровень мотивации)
4	Обучающийся 4	59	мотивация ярко выраженная (высокий уровень мотивации)
5	Обучающийся 5	61	мотивация ярко выраженная (высокий уровень мотивации)
6	Обучающийся 6	49	мотивация достаточно высокая (уровень мотивации выше среднего)
7	Обучающийся 7	54	мотивация достаточно высокая (уровень мотивации выше среднего)
8	Обучающийся 8	48	мотивация достаточно высокая (уровень мотивации выше среднего)
9	Обучающийся 9	60	мотивация ярко выраженная (высокий уровень мотивации)
10	Обучающийся 10	58	мотивация достаточно высокая (уровень мотивации выше среднего)

Анализируя уровень мотивации учащихся на констатирующем и контрольном этапах эксперимента, мы выявили положительную динамику, что отражено на рисунке 2. Этот результат свидетельствует о том, что ученики проявляют все больший интерес и вовлеченность в учебный процесс.

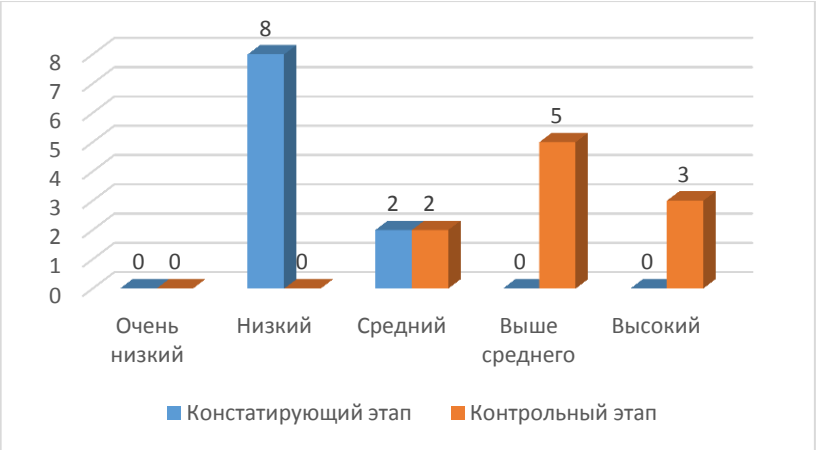


Рисунок 2 – Сравнение уровня мотивации учащихся на констатирующем и контрольном этапах эксперимента

Исходя из представленной диаграммы, можно сделать вывод, что на контрольном этапе произошло значительное увеличение уровня мотивации учащихся. Отмечается, что в настоящее время не наблюдается ни одного учащегося с очень низким или низким уровнем мотивации. Перед экспериментом, ни один из учащихся не имел среднего, выше среднего или высокого уровней мотивации. Однако после эксперимента, двое учащихся достигли среднего уровня мотивации, пять учащихся достигли уровня выше среднего, и трое учащихся достигли высокого уровня мотивации. Это указывает на положительные результаты эксперимента, который способствовал повышению уровня мотивации учащихся.

Для подтверждения различий в уровнях мотивации до и после проведения эксперимента, мы обратимся к статистическому инструменту Т-критерию Вилкоксона [13]. Этот метод позволяет нам определить надежность различий, выявленных в наших данных.

Сформулируем рабочие гипотезы исследования, с целью детального анализа и выявления факторов, которые могут иметь влияние на предмет нашего исследования.

H_0: повышение уровня мотивации к изучению программирования у обучающихся АНО «Центр информационных технологий «РазвИТие» в результате освоения учебного курса «Основы программирования на Python» не является статистически значимым.

H_1: повышение уровня мотивации к изучению программирования у обучающихся АНО «Центр информационных технологий «РазвИТие» в результате освоения учебного курса «Основы программирования на Python» является статистически значимым.

Для выполнения вычисления Т-критерия Вилкоксона необходимо начать с вычитания каждого индивидуального значения «до» из значения «после». Операция вычитания значений «констатирующего этапа» из значений «контрольного этапа» представлена в таблице 4.

Таблица 4 – Первый шаг подсчета Т-критерия Вилкоксона

№ п/п	Констатирующий этап, $t_{до}$	Контрольный этап, $t_{после}$	Разность, ($t_{до} – t_{после}$)	Абсолютное значение разности
1	25	45	20	20
2	30	50	20	20
3	23	41	18	18
4	35	59	24	24
5	28	61	33	33
6	27	49	22	22
7	21	54	33	33
8	29	48	19	19
9	32	60	28	28
10	24	58	34	34

Следующим шагом в процессе является исключение нулевых сдвигов, однако в данном случае нам необходимо учитывать, что в нашей матрице присутствуют связанные ранги – ряды с одинаковыми ранговыми номерами. Исходя из этого, необходимо осуществить переформирование этих рядов.

Переформирование рангов осуществляется с сохранением важности каждого ранга. Таким образом, между ранговыми номерами должны оставаться соответствующие соотношения – некоторые ранги должны быть больше, некоторые меньше, а некоторые могут быть равны друг другу. Однако не рекомендуется назначать ранги выше 1 или ниже значения, соответствующего количеству параметров (в данном случае n = 10).

Конкретное переформирование рангов осуществляется в таблице 5, которая предоставляет соответствующую информацию по присвоению новых рангов каждому ряду.

Таблица 5 – Переформирование рангов

Номера мест в упорядоченном ряду	Расположение факторов по оценке эксперта	Новые ранги
1	18	1
2	19	1.6
3	20	2.1
4	20	2.1

Продолжение таблицы 5

5	22	3.2
6	24	4.4
7	28	6.6
8	33	9.4
9	33	9.4
10	34	10

Следующим этапом в расчете Т-критерия является вычисление рангового номера разности, как указано в таблице 6.

Таблица 6 – Ранговый номер разности

До измерения, $t_{до}$	После измерения, $t_{после}$	Разность ($t_{до}$-$t_{после}$)	Абсолютное значение разности	Ранговый номер разности
25	45	20	20	2.1
30	50	20	20	2.1
23	41	18	18	1
35	59	24	24	4.4
28	61	33	33	9.4
27	49	22	22	3.2
21	54	33	33	9.4
29	48	19	19	5.5
32	60	28	28	6.6
24	58	34	34	10
Сумма				53.7

Сумма по столбцу рангов равна ∑=53,7.

Далее нужно проверить правильно ли была составлена матрица с использованием подсчета по формуле (1), контрольной суммы:

$$\sum \boldsymbol{x_{ij}} = \frac{\boldsymbol{(1+n)n}}{\boldsymbol{2}} = \frac{\boldsymbol{(1+10)10}}{\boldsymbol{2}} = \boldsymbol{53,7} \qquad (1)$$

Так как мы видим, что контрольная сумма равна сумме по столбцу, то можно сделать вывод, что ранжирование было сделано верно.

Следующим шагом мы должны указать на нетипичные направления, то есть в нашем случае – отрицательные. Ранговая сумма таких нетипичных направлений образует эмпирическое значение критерия Т по формуле (2):

$$\mathbf{T} = \sum \boldsymbol{R_t} = \mathbf{0} \quad (2)$$

Далее ищем критические значения для Т-критерия Вилкоксона для n=10 (Таблица 7):

Таблица 7 – Критические значения для Т-критерия Вилкоксона

n	$T_{кр}$	
	p≤0.05	p≤0.01
10	10	5

Для того чтобы сделать вывод, изобразим на оси значимости Тэмп (рис. 3).

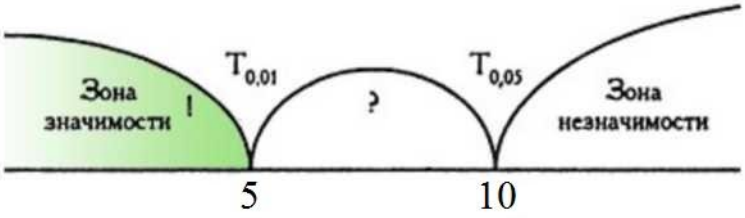


Рисунок 3 – Ось значимости

В нашей ситуации эмпирическое значение Т входит в зону значимости: $T_{эмп}<T_{кр}(0,01)$.

Проведенный эксперимент позволяет сделать вывод о значимости и неслучайности изменений, выявленных в ходе исследования. Объективный анализ данных позволяет установить, что полученные результаты имеют определенную статистическую значимость, что свидетельствует о достоверности и надежности полученных выводов.

Гипотеза H_1 принимается. Повышение уровня мотивации к изучению программирования у обучающихся АНО «Центр информационных технологий «РазвИТие» в результате освоения учебного курса «Основы программирования на Python» является статистически значимым.

Таким образом, сформулированная нами гипотеза о том, что, если при обучению программированию на языке программирования Python в системе дополнительного образования применить практико-ориентированный подход, то произойдет повышение уровня мотивации обучающихся к изучению программирования подтвердилась.

Выводы по главе 3

Целью экспериментальной работы являлось доказать эффективность применения методики обучения языку программирования Python в системе дополнительного образования для повышения мотивации обучающихся к изучению программирования.

Экспериментальная работа осуществлялась в три этапа с 2023 по 2024 год на базе АНО «Центр информационных технологий «РазвИТие».

На первом этапе исследования был проведен анализ уровня мотивации учащихся на основе заранее разработанных критериев. Результаты показали, что у большинства обучающихся преобладает низкий уровень мотивации. Этот факт вызывает серьезную озабоченность и является одной из ключевых проблем, которую необходимо было решить в рамках нашего исследования.

При проведении второго этапа происходило внедрение разработанной методики учебного курса «Основы программирования на Python» с целью повышения мотивации школьников к изучению программированию.

На третьем этапе исследования был проведен детальный эксперимент, который в свою очередь позволил сделать ряд важных выводов относительно процесса развития повышения мотивации школьников к изучению программированию, а именно, что мотивация к изучению программированию будет выше, если использовать для обучения практико-ориентированный подход.

ЗАКЛЮЧЕНИЕ

В ходе исследования была сформулирована гипотеза о том, что использование практико-ориентированного подхода будет более эффективным для повышения мотивации школьников к изучению программирования.

В теоретической части работы был проведен анализ нормативно-правовой документации и психолого-педагогической литературы, который показал актуальность повышения мотивации школьников к изучению программирования в современном обществе. Организация учебного процесса по предмету информатика с помощью современных учебников уже вносит базовую подготовку учащихся, но требуется актуализация и дополнение материалов с упором на практическую составляющую.

Анализ теоретических материалов показал, что практико-ориентированный подход является одной из самых современных и актуальных технологий обучения. Он предоставляет возможность ученикам самостоятельно изучать, разбирать и закреплять как теоретические, так и практические знания, и навыки.

В практической части работы было проведено подтверждение сформулированной гипотезы. Сначала было изучено содержание школьного курса информатики с фокусом на программирование, а затем была разработана методика обучения языку программирования Python в системе дополнительного образования.

Заключительным этапом исследования стал педагогический эксперимент, который подтвердил гипотезу о том, что использование практико-ориентированного подхода эффективнее для повышения мотивации школьников к изучению программирования. Это подтверждение было получено с помощью методов математической статистики.

Цель работы была достигнута, а поставленные задачи выполнены. Можно сделать вывод о верности сформулированной гипотезы.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Баклыкова Ю. В. Развитие мотивации у школьников на уроках информатики с помощь конкурсов и олимпиад / Ю. В. Баклыкова, А. И. Ишмаева // Математика, информатика, физика: проблемы и перспективы : сборник научных статей международной научно-практической конференции, Оренбург, 21–22 апреля 2022 года / ФГБОУ ВО «Оренбургский государственный педагогический университет». – Оренбург, 2022. – С. 261-269.

2. Белоусова Ю. В. Мотивация учащихся общеобразовательных школ к изучению программирования / Ю.В. Белоусова // Калининградский вестник образования. – 2023. – № 3 (19). – С. 77-84.

3. Беляева М. Б. Методические рекомендации по использованию занимательных задач на уроках информатики / М.Б. Беляева, Е.А. Ильясова, Е.А. Калякина, М.М. Савинкова // Вестник педагогических наук. – 2023. – № 7. – С. 168-174.

4. Босова Л. Л. Информатика. 7–9 классы: примерная рабочая программа / Л.Л. Босова, А. Ю. Босова – Москва : БИНОМ. Лаборатория знаний, 2016. – 30 с.

5. Булатова А. А. Применение интерактивного метода обучения на уроках информатики / А.А. Булатова, И.Н. Смирнова // Естественные, математические и технические науки. Образование. Технологии. Инновации. Материалы Межрегиональной научно-практической студенческой конференции. Липецк. – 2023. – С. 53-55.

6. Вальков Н. А. Разработка компьютерных игр как мотивация к изучению алгоритмизации и программирования в основной школе / Н.А. Вальков // Ломоносовские научные чтения студентов, аспирантов и молодых учёных высшей школы естественных наук и технологий САФУ – 2022. – 2022. – С. 18-22.

7. Волкова Е. Н. Внутренняя мотивация деятельности современной молодежи как условие высоких достижений в сфере программирования: к вопросу определения понятия / Е.Н. Волкова, О.М. Исаева, А.Н. Морева // Вестник Мининского университета. – 2022. – Т. 10. – № 2 (39).

8. Гриншкун В. В. Разработка подходов к подготовке учителей к применению цифровых ресурсов для построения индивидуальных образовательных траекторий с учётом личностных особенностей школьников / В.В. Гриншкун // Информатизация образования и методика электронного обучения: цифровые технологии в образовании. Материалы V Международной научной конференции. Красноярск. – 2021. – С. 85-89.

9. Евстифеев Н. О. Проблема мотивации в обучении программированию учащихся основной школы / Н.О. Евстифеев // В сборнике: Роль психолого-педагогических исследований в инновационном развитии общества. Сборник статей Международной научно-практической конференции. – 2019. – С. 58-61.

10. Евстифеев Н. О. Результаты исследования мотивации школьников к обучению программированию посредством робототехники / Н.О. Евстифеев // Инновационные психологические и педагогические технологии как механизм повышения качества образования. Сборник статей Международной научно-практической конференции. – 2020. – С. 47-48.

11. Капитанов А. И. Повышение мотивации обучающихся за счет применения соревновательного подхода при изучении языка программирования Python / А.И. Капитанов, И.И. Капитанова // Системы компьютерной математики и их приложения. – 2023. – № 24. – С. 390-395.

12. Кожанова А. М. Преподавание программирования в классах с углубленным изучением информатики / А. М. Кожанова, Т. И. Крылова // Физико-математическое и естественно-научное образование: наука и школа: Материалы Всероссийской научно-практической конференции преподавателей высшей и средней школы, Йошкар-Ола, 29 апреля 2022

года. – Йошкар-Ола: Марийский государственный университет, 2022. – С. 134-139.

13. Коченгин А. Е. Непараметрический критерий Вилкоксона как элемент процедуры метода распознавания критических событий / А. Е. Коченгин, В. А. Шихин // Прикладная информатика. – 2020. – № 2 (86). – С. 74–88.

14. Литвинов В. А. О повышении мотивации к обучению информатике / В.А. Литвинов // Вестник УЮИ. – 2020. – №2 (88).

15. Мардер И. А. Повышение мотивации школьников при обучении программированию / И.А. Мардер, И.В. Шимов // Актуальные вопросы преподавания математики, информатики и информационных технологий. – 2023. – № 8. – С. 166-171.

16. Маркелов В. К. Игровые диалоговые программы как инструмент мотивации к изучению программирования в школе / В.К. Маркелов, О.А. Завьялова // Современные тренды образования. Материалы IV Всероссийской (национальной) педагогической научно-практической конференции. Ивановский государственный университет, Шуйский филиал. Шуя. – 2022. – С. 81-89.

17. Маркелов В. К. Преимущества использования языка Python при обучении программированию в школьном курсе информатики / В. К. Маркелов, О. А. Завьялова // Современные тренды образования: Материалы V Всероссийской (национальной) педагогической научно-практической конференции, Шуя, 15–16 декабря 2022 года. Шуйский филиал федерального государственного бюджетного образовательного учреждения высшего образования «Ивановский государственный университет». Шуя. – 2023. – С. 77-81.

18. Мункоев А. А. Повышение мотивации к изучению программирования у учащихся старших классов / А.А. Мункоев // Вопросы педагогики. – 2021. – № 1-1. – С. 173-176.

19. Паршукова Н. Б. Методические аспекты преподавания нескольких языков программирования для будущих IT-специалистов / Н. Б. Паршукова // Современные тенденции естественно-математического образования: материалы XII Всероссийской научно-практической конференции с международным участием, Соликамск, 07–08 апреля 2023 года. – Соликамск: Соликамский государственный педагогический институт. – 2023. – С. 71-74.

20. Перегуда А. В Стимулирование познавательной и творческой активности обучающихся на уроках информатики / А.В. Перегуда // Вопросы педагогики. – 2020. – № 9-2. – С. 202-206.

21. Сайдуллина Е. В. Использование различных интегрированных сред разработки при обучении с целью повышения мотивации обучающихся к изучению программирования / Е. В. Сайдуллина // Научный аспект. – 2024. – №2-2024(02/24-06-056).

22. Сайдуллина Е. В. Популяризация мультипарадигмального языка программирования python при подготовке к сдаче ЕГЭ по информатике / Е. В. Сайдуллина // Научный аспект. – 2024. – №2-2024(02/24-06-057).

23. Соколов Н. К. Повышение мотивации детей к изучению программирования с использованием игровых методов / Н.К. Соколов, А.Д. Емельянов // Ресурсам области – эффективное использование. Сборник: материалов XIX Ежегодной научной конференции студентов Технологического университета. – 2019. – С. 27-34.

24. Соколова А. Н. Сюжетные задачи как средство мотивации школьников при изучении основ программирования / А.Н. Соколова, Н.В. Шалагинова // Проблемы теории и практики инновационного развития и интеграции современной науки и образования. Материалы III Международной междисциплинарной конференции. – 2022. – С. 178-184.

25. Щедрина И. О. Нарратив, интерес, мотивация: Л. И. Петражицкий VS Я. Э. Голосовкер / И.О. Щедрина // Вестник Санкт-

Петербургского университета. Философия и конфликтология. – 2023. – Т. 39. – № 1. – С. 81-91.

ПРИЛОЖЕНИЕ

Приложение 1

Конспект занятия по теме «Множественное ветвление: if-elif-else»

Ранее мы рассмотрели работу условного оператора if. С помощью его расширенной версии if-else можно реализовать две отдельные ветви выполнения. Однако алгоритм программы может предполагать выбор больше, чем из двух путей, например, из трех, четырех или даже пяти. В данном случае следует говорить о необходимости множественного ветвления.

Рассмотрим конкретный пример. Допустим, в зависимости от возраста пользователя, ему рекомендуется определенный видеоконтент. При этом выделяют группы от 3 до 6 лет, от 6 до 12, от 12 до 16, 16+. Итого 4 диапазона. Как бы мы стали реализовывать задачу, имея в наборе инструментов только конструкцию if-else?

Самый простой ответ – последовательно проверять вхождение введенного числа-возраста в определенный диапазон с помощью следующих друг за другом условных операторов:

```
old = int(input('Ваш возраст: '))

print('Рекомендовано:')

if 3 <= old < 6:
    print('"Заяц в лабиринте"')

if 6 <= old < 12:
    print('"Марсианин"')

if 12 <= old < 16:
    print('"Загадочный остров"')

if 16 <= old:
    print('"Поток сознания"')
```

Примечание. Названия фильмов выводятся на экран в двойных кавычках. Поэтому в программе для определения строк используются одинарные.

Предложенный код прекрасно работает, но есть одно существенное "но". Он не эффективен, так как каждый if в нем – это отдельно взятый оператор, никак не связанный с другими if. Процессор тратит время и "нервы" на обработку каждого из них, даже если в этом уже нет необходимости. Например, введено число 10. В первом if логическое выражение возвращает ложь, и поток выполнения переходит ко второму if. Логическое выражение в его заголовке возвращает истину, и его тело выполняется. Всё, на этом программа должна была остановиться.

Однако следующий if никак не связан с предыдущим, поэтому далее будет проверяться вхождение значения переменной old в диапазон от 12 до 16, в чем необходимости нет. И далее будет обрабатываться логическое выражение в последнем if, хотя уже понятно, что и там будет False. Что же делать?

Ответом является вложение условных операторов друг в друга:

```
old = int(input('Ваш возраст: '))

print('Рекомендовано:')

if 3 <= old < 6:
    print('"Заяц в лабиринте"')
else:
    if 6 <= old < 12:
        print('"Марсианин"')
    else:
        if 12 <= old < 16:
            print('"Загадочный остров"')
        else:
            if 16 <= old:
                print('"Поток сознания"')
```

Рассмотрим поток выполнения этого варианта кода. Сначала проверяется условие в первом if (он же самый внешний). Если здесь было получено True, то тело этого if выполняется, а в ветку else мы даже не заходим, так как она срабатывает только тогда, когда в условии if возникает ложь.

Если внешний if вернул False, поток выполнения программы заходит в соответствующий ему внешний else. В его теле находится другой if со своим else. Если введенное число попадает в диапазон от 6 до 12, то выполнится тело вложенного if, после чего программа завершается. Если же число не попадает в диапазон от 6 до 12, то произойдет переход к ветке else. В ее теле находится свой условный оператор, имеющий уже третий уровень вложенности.

Таким образом до последней проверки (16 <= old) интерпретатор доходит только тогда, когда все предыдущие возвращают False. Если же по ходу выполнения программы возникает True, то все последующие проверки опускаются, что экономит ресурсы процессора. Кроме того, такая логика выполнения программы более правильная.

Теперь зададимся следующим вопросом. Можно ли как-то оптимизировать код множественного ветвления и не строить лестницу из вложенных друг в друга условных операторов? Во многих языках программирования, где отступы используются только для удобства чтения программистом, но не имеют никакого синтаксического значения, часто используется подобный стиль:

```
if логическое_выражение {
```

```
    … ;
}
else if логическое_выражение {
    … ;
}
else if логическое_выражение {
    … ;
}
else {
    … ;
}
```

Может показаться, что имеется только один уровень вложенности, и появляется новое расширение для if, выглядящее как else if. Но это только кажется. На самом деле if, стоящее сразу после else, является вложенным в это else. Выше приведенная схема – то же самое, что

```
if логическое_выражение {
    … ;
}
else
    if логическое_выражение {
        … ;
    }
    else
        if логическое_выражение {
            … ;
        }
        else {
            … ;
        }
```

Именно так ее "понимает" интерпретатор или компилятор. Однако считается, что человеку проще воспринимать первый вариант.

В Питоне подобный номер с поднятием вложенного if к более внешнему else не сработает, потому что в нем отступы и переходы на новую строку имеют синтаксическое значение. Поэтому в язык Python встроена возможность настоящего **множественного ветвления на одном уровне вложенности, которое реализуется с помощью веток elif**.

Слово "elif" образовано от двух первых букв слова "else", к которым присоединено слово "if". Это можно перевести как "иначе если".

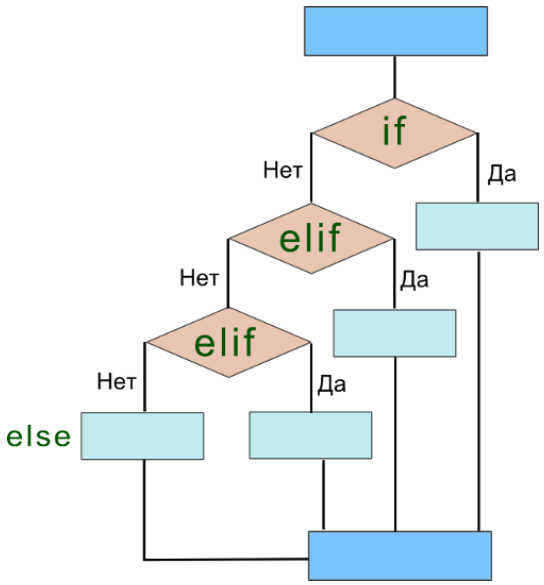
В отличие от else, **в заголовке elif обязательно должно быть логическое выражение** также, как в заголовке if. Перепишем нашу программу, используя конструкцию множественного ветвления:

```
old = int(input('Ваш возраст: '))

print('Рекомендовано:')
```

```
if 3 <= old < 6:
    print('''Заяц в лабиринте''')
elif 6 <= old < 12:
    print('''Марсианин''')
elif 12 <= old < 16:
    print('''Загадочный остров''')
elif 16 <= old:
    print('''Поток сознания''')
```

Обратите внимание, в конце, после всех elif, может использоваться одна ветка else для обработки случаев, не попавших в условия ветки if и всех elif. Блок-схему полной конструкции if-elif-…-elif-else можно изобразить так:



Как только тело if или какого-нибудь elif выполняется, программа сразу же возвращается в основную ветку (нижний ярко-голубой прямоугольник), а все нижеследующие elif, а также else пропускаются.

Практическая работа

Напишите программу, которая запрашивает на ввод число. Если оно положительное, то на экран выводится цифра 1. Если число отрицательное, выводится -1. Если введенное число – это 0, то на экран выводится 0. Используйте в коде условный оператор множественного ветвления.

Конспект занятия по теме «Циклы в программировании. Цикл while.»

Циклы являются такой же важной частью структурного программирования, как условные операторы. С помощью циклов можно организовать повторение выполнения участков кода. Потребность в этом возникает довольно часто. Например, пользователь последовательно вводит числа, и каждое из них требуется добавлять к общей сумме. Или нужно вывести на экран квадраты ряда натуральных чисел и тому подобные задачи.

Цикл while

"While" переводится с английского как "пока". Но не в смысле "до

свидания", а в смысле "пока имеем это, делаем то".

Можно сказать, while является универсальным циклом. Он присутствует во всех языках, поддерживающих структурное программирование, в том числе в Python. Его синтаксис обобщенно для всех языков можно выразить так:

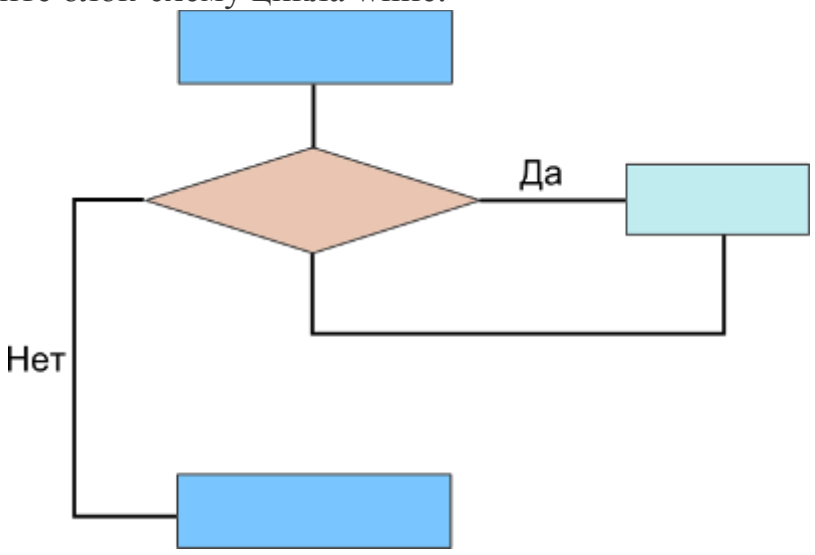
```
while логическое_выражение {
   выражение 1;
   …
   выражение n;
}
```

Это похоже на условный оператор if. Однако в случае циклических операторов их тела могут выполняться далеко не один раз. В случае if, если логическое выражение в заголовке возвращает истину, то тело выполняется единожды. После этого поток выполнения программы возвращается в основную ветку и выполняет следующие выражения, расположенные ниже всей конструкции условного оператора.

В случае while, после того как его тело выполнено, поток возвращается к заголовку цикла и снова проверяет условие. Если логическое выражение возвращает истину, то тело снова выполняется. Потом снова возвращаемся к заголовку и так далее.

Цикл завершает свою работу только тогда, когда логическое выражение в заголовке возвращает ложь, то есть условие выполнения цикла больше не соблюдается. После этого поток выполнения перемещается к выражениям, расположенным ниже всего цикла. Говорят, "происходит выход из цикла".

Рассмотрите блок-схему цикла while.



На ней ярко-голубыми прямоугольниками обозначена основная ветка программы, ромбом – заголовок цикла с логическим выражением, бирюзовым прямоугольником – тело цикла.

С циклом while возможны две исключительные ситуации:

- Если при первом заходе в цикл логическое выражение возвращает False, то тело цикла не выполняется ни разу. Эту ситуацию можно считать

нормальной, так как при определенных условиях логика программы может предполагать отсутствие необходимости в выполнении выражений тела цикла.

- Если логическое выражение в заголовке while никогда не возвращает False, а всегда остается равным True, то цикл никогда не завершится, если только в его теле нет оператора принудительного выхода из цикла (break) или вызовов функций выхода из программы – quit(), exit() в случае Python. Если цикл повторяется и повторяется бесконечное количество раз, то в программе происходит **зацикливание**. В это время она зависает и самостоятельно завершиться не может.

Вспомним наш пример из урока про исключения. Пользователь должен ввести целое число. Поскольку функция input() возвращает строку, то программный код должен преобразовать введенное к целочисленному типу с помощью функции int(). Однако, если были введены символы, не являющиеся цифрами, то возникает исключение ValueError, которое обрабатывается веткой except. На этом программа завершается.

Другими словами, если бы программа предполагала дальнейшие действия с числом (например, проверку на четность), а она его не получила, то единственное, что программа могла сделать, это закончить свою работу досрочно.

Но ведь можно просить и просить пользователя корректно вести число, пока он его не введет. Вот как может выглядеть реализующий это код:

```
n = input("Введите целое число: ")

while type(n) != int:
    try:
        n = int(n)
    except ValueError:
        print("Неправильно ввели!")
        n = input("Введите целое число: ")

if n % 2 == 0:
    print("Четное")
else:
    print("Нечетное")
```

Примечание 1. Не забываем, в языке программирования Python в конце заголовков сложных инструкций ставится двоеточие.

Примечание 2. В выражении type(n) != int с помощью функции type() проверяется тип переменной n. Если он не равен int, т. е. значение n не является целым числом, а является в данном случае строкой, то выражение возвращает истину. Если же тип n равен int, то данное логическое выражение возвращает ложь.

Примечание 3. Оператор % в языке Python используется для нахождения остатка от деления. Так, если число четное, то оно без остатка

делится на 2, т. е. остаток будет равен нулю. Если число нечетное, то остаток будет равен единице.

Проследим алгоритм выполнения этого кода. Пользователь вводит данные, они имеют строковый тип и присваиваются переменной n. В заголовке while проверяется тип n. При первом входе в цикл тип n всегда строковый, т. е. он не равен int. Следовательно, логическое выражение возвращает истину, что позволяет зайти в тело цикла.

Здесь в ветке try совершается попытка преобразования строки к целочисленному типу. Если она была удачной, то ветка except пропускается, и поток выполнения снова возвращается к заголовку while.

Теперь n связана с целым числом, следовательно, ее тип int, который не может быть не равен int. Он ему равен. Таким образом логическое выражение type(n) != int возвращает False, и весь цикл завершает свою работу. Далее поток выполнения переходит к оператору if-else, находящемуся в основной ветке программы. Здесь могло бы находиться что угодно, не обязательно условный оператор.

Вернемся назад. Если в теле try попытка преобразования к числу была неудачной, и было выброшено исключение ValueError, то поток выполнения программы отправляется в ветку except и выполняет находящиеся здесь выражения, последнее из которых просит пользователя снова ввести данные. Переменная n теперь имеет новое значение.

После завершения except снова проверяется логическое выражение в заголовке цикла. Оно даст True, т. к. значение n по-прежнему строка.

Выход из цикла возможен только тогда, когда значение n будет успешно конвертировано в число.

Рассмотрим следующий пример:

```
total = 100

i = 0
while i < 5:
    n = int(input())
    total = total - n
    i = i + 1

print("Осталось", total)
```

Сколько раз "прокрутится" цикл в этой программе, т. е. сколько итераций он сделает? Ответ: 5.

1. Сначала переменная i равна 0. В заголовке цикла проверяется условие i < 5, и оно истинно. Тело цикла выполняется. В нем меняется значение i, путем добавления к нему единицы.
2. Теперь переменная i равна 1. Это меньше пяти, и тело цикла выполняется второй раз. В нем i меняется, ее новое значение 2.
3. Два меньше пяти. Тело цикла выполняется третий раз. Значение i становится равным трем.
4. Три меньше пяти. На этой итерации i присваивается 4.

5. Четыре по прежнему меньше пяти. К i добавляется единица, и теперь ее значение равно пяти.

Далее начинается шестая итерация цикла. Происходит проверка условия i < 5. Но поскольку теперь оно возвращает ложь, то выполнение цикла прерывается, и его тело не выполняется.

"Смысловая нагрузка" данного цикла – это последовательное вычитание из переменной total вводимых чисел. Переменная i в данном случае играет только роль счетчика итераций цикла. В других языках программирования для таких случаев предусмотрен цикл for, который так и называется: "цикл со счетчиком". Его преимущество заключается в том, что в теле цикла не надо изменять переменную-счетчик, ее значение меняется автоматически в заголовке for.

В языке Python тоже есть цикл for. Но это не цикл со счетчиком. В Питоне он предназначен для перебора элементов последовательностей и других сложных объектов. Данный цикл и последовательности будут изучены в последующих уроках.

Для while наличие счетчика не обязательно. Представим, что надо вводить числа, пока переменная total больше нуля. Тогда код будет выглядеть так:

```
total = 100

while total > 0:
    n = int(input())
    total = total - n

print("Ресурс исчерпан")
```

Сколько раз здесь выполнится цикл? Неизвестно, все зависит от вводимых значений. Поэтому у цикла со счетчиком известно количество итераций, а у цикла без счетчика – нет.

Самое главное для цикла while – чтобы в его теле происходили изменения значений переменных, которые проверяются в его заголовке, и чтобы хоть когда-нибудь наступил случай, когда логическое выражение в заголовке возвращает False. Иначе произойдет зацикливание.

Примечание 1. Не обязательно в выражениях total = total - n и i = i + 1 повторять одну и ту же переменную. В Python допустим сокращенный способ записи подобных выражений: total -= n и i += 1.

Примечание 2. При использовании счетчика он не обязательно должен увеличиваться на единицу, а может изменяться в любую сторону на любое значение. Например, если надо вывести числа кратные пяти от 100 до 0, то изменение счетчика будет таким i = i - 5, или i -= 5.

Примечание 3. Для счетчика не обязательно использовать переменную с идентификатором i. Можно назвать переменную-счетчик как угодно. Однако так принято в программировании, что счетчики обозначают именами i и j (иногда одновременно требуются два счетчика).

Практическая работа

Измените последний код из урока так, чтобы переменная total не могла уйти в минус. Например, после предыдущих вычитаний ее значение стало равным 25. Пользователь вводит число 30. Однако программа не выполняет вычитание, а выводит сообщение о недопустимости операции, после чего осуществляет выход из цикла.

Конспект занятия по теме «Функции в программировании.»

Функция в программировании представляет собой обособленный участок кода, который можно вызывать, обратившись к нему по имени, которым он был назван. При вызове происходит выполнение команд тела функции.

Функции можно сравнить с небольшими программками, которые сами по себе, т. е. автономно, не исполняются, а встраиваются в обычную программу. Нередко их так и называют – подпрограммы. Других ключевых отличий функций от программ нет. Функции также при необходимости могут получать и возвращать данные. Только обычно они их получают не с ввода (клавиатуры, файла и др.), а из вызывающей программы. Сюда же они возвращают результат своей работы.

Существует множество встроенных в язык программирования функций. С некоторыми такими в Python мы уже сталкивались. Это print(), input(), int(), float(), str(), type(). Код их тела нам не виден, он где-то "спрятан внутри языка". Нам же предоставляется только интерфейс – имя функции.

С другой стороны, программист всегда может определять свои функции. Их называют пользовательскими. В данном случае под "пользователем" понимают программиста, а не того, кто пользует программу. Разберемся, зачем нам эти функции, и как их создавать.

Предположим, надо три раза подряд запрашивать на ввод пару чисел и складывать их. С этой целью можно использовать цикл:

```
i = 0
while i < 3:
    a = int(input())
    b = int(input())
    print(a+b)
    i += 1
```

Однако, что если перед каждым запросом чисел, надо выводить надпись, зачем они нужны, и каждый раз эта надпись разная. Мы не можем прервать цикл, а затем вернуться к тому же циклу обратно. Придется отказаться от него, и тогда получится длинный код, содержащий в разных местах одинаковые участки:

```
print("Сколько бананов и ананасов для обезьян?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
```

```
print("Сколько жуков и червей для ежей?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")

print("Сколько рыб и моллюсков для выдр?")
a = int(input())
b = int(input())
print("Всего", a+b, "шт.")
```

Пример исполнения программы:

```
Сколько бананов и ананасов для обезьян?
15
5
Всего 20 шт.
Сколько жуков и червей для ежей?
50
12
Всего 62 шт.
Сколько рыб и моллюсков для выдр?
16
8
Всего 24 шт.
```

Внедрение функций позволяет решить проблему дублирования кода в разных местах программы. Благодаря им можно исполнять один и тот же участок кода не сразу, а только тогда, когда он понадобится.

Определение функции. Оператор def

В языке программирования Python функции определяются с помощью оператора def. Рассмотрим код:

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
```

Это пример определения функции. Как и другие сложные инструкции вроде условного оператора и циклов функция состоит из заголовка и тела. Заголовок оканчивается двоеточием и переходом на новую строку. Тело имеет отступ.

Ключевое слово def сообщает интерпретатору, что перед ним определение функции. За def следует имя функции. Оно может быть любым, также как и всякий идентификатор, например, переменная. В программировании весьма желательно давать всему осмысленные имена. Так в данном случае функция названа "посчитатьЕду" в переводе на русский.

После имени функции ставятся скобки. В приведенном примере они пустые. Это значит, что функция не принимает никакие данные из вызывающей ее программы. Однако она могла бы их принимать, и тогда в

скобках были бы указаны так называемые параметры.

После двоеточия следует тело, содержащее инструкции, которые выполняются при вызове функции. Следует различать определение функции и ее вызов. В программном коде они не рядом и не вместе. Можно определить функцию, но ни разу ее не вызвать. Нельзя вызвать функцию, которая не была определена. Определив функцию, но ни разу не вызвав ее, вы никогда не выполните ее тела.

Вызов функции

Рассмотрим полную версию программы с функцией:

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")

print("Сколько бананов и ананасов для обезьян?")
countFood()

print("Сколько жуков и червей для ежей?")
countFood()

print("Сколько рыб и моллюсков для выдр?")
countFood()
```

После вывода на экран каждого информационного сообщения осуществляется вызов функции, который выглядит просто как упоминание ее имени со скобками. Поскольку в функцию мы ничего не передаем скобки опять же пустые. В приведенном коде функция вызывается три раза.

Когда функция вызывается, поток выполнения программы переходит к ее определению и начинает исполнять ее тело. После того, как тело функции исполнено, поток выполнения возвращается в основной код в то место, где функция вызывалась. Далее исполняется следующее за вызовом выражение.

В языке Python определение функции должно предшествовать ее вызовам. Это связано с тем, что интерпретатор читает код строка за строкой и о том, что находится ниже по течению, ему еще неизвестно. Поэтому если вызов функции предшествует ее определению, то возникает ошибка (выбрасывается исключение NameError):

```
print("Сколько бананов и ананасов для обезьян?")
countFood()

print("Сколько жуков и червей для ежей?")
countFood()

print("Сколько рыб и моллюсков для выдр?")
countFood()
```

```
def countFood():
    a = int(input())
    b = int(input())
    print("Всего", a+b, "шт.")
```

Результат:

```
Сколько бананов и ананасов для обезьян?
Traceback (most recent call last):
  File "test.py", line 2, in <module>
    countFood()
NameError: name 'countFood' is not defined
```

Для многих компилируемых языков это не обязательное условие. Там можно определять и вызывать функцию в произвольных местах программы. Однако для удобочитаемости кода программисты даже в этом случае предпочитают соблюдать определенные правила.

Функции придают программе структуру

Польза функций не только в возможности многократного вызова одного и того же кода из разных мест программы. Не менее важно, что благодаря им программа обретает истинную структуру. Функции как бы разделяют ее на обособленные части, каждая из которых выполняет свою конкретную задачу.

Пусть надо написать программу, вычисляющую площади разных фигур. Пользователь указывает, площадь какой фигуры он хочет вычислить. После этого вводит исходные данные. Например, длину и ширину в случае прямоугольника. Чтобы разделить поток выполнения на несколько ветвей, следует использовать оператор if-elif-else:

```
figure = input("1-прямоугольник, 2-треугольник, 3-круг: ")

if figure == '1':
  a = float(input("Ширина: "))
  b = float(input("Высота: "))
  print("Площадь: %.2f" % (a*b))
elif figure == '2':
  a = float(input("Основание: "))
  h = float(input("Высота: "))
  print("Площадь: %.2f" % (0.5 * a * h))
elif figure == '3':
  r = float(input("Радиус: "))
  print("Площадь: %.2f" % (3.14 * r**2))
else:
  print("Ошибка ввода")
```

Здесь нет никаких функций, и все прекрасно. Но напишем вариант с функциями:

```
def rectangle():
    a = float(input("Ширина: "))
```

```
    b = float(input("Высота: "))
    print("Площадь: %.2f" % (a*b))

def triangle():
    a = float(input("Основание: "))
    h = float(input("Высота: "))
    print("Площадь: %.2f" % (0.5 * a * h))

def circle():
    r = float(input("Радиус: "))
    print("Площадь: %.2f" % (3.14 * r**2))

figure = input("1-прямоугольник, 2-треугольник, 3-круг: ")
if figure == '1':
  rectangle()
elif figure == '2':
  triangle()
elif figure == '3':
  circle()
else:
  print("Ошибка ввода")
```

Он кажется сложнее, а каждая из трех функций вызывается всего один раз. Однако, из общей логики программы как бы убраны и обособлены инструкции для нахождения площадей. Программа теперь состоит из отдельных "кирпичиков Лего". В основной ветке мы можем комбинировать их как угодно. Она играет роль управляющего механизма.

Если нам когда-нибудь захочется вычислять площадь треугольника по формуле Герона, а не через высоту, то не придется искать код во всей программе (представьте, что она состоит из тысяч строк кода как реальные программы). Мы пойдем к месту определения функций и изменим тело одной из них.

Если понадобиться использовать эти функции в какой-нибудь другой программе, то мы сможем импортировать их туда, сославшись на данный файл с кодом (как это делается в Python, будет рассмотрено позже).

Практическая работа

В программировании можно из одной функции вызывать другую. Для иллюстрации этой возможности напишите программу по следующему описанию.

Основная ветка программы, не считая заголовков функций, состоит из одной строки кода. Это вызов функции test(). В ней запрашивается на ввод целое число. Если оно положительное, то вызывается функция positive(), тело которой содержит команду вывода на экран слова "Положительное". Если число отрицательное, то вызывается функция negative(), ее тело содержит выражение вывода на экран слова "Отрицательное".

Понятно, что вызов test() должен следовать после определения функций. Однако имеет ли значение порядок определения самих функций? То есть должны ли определения positive() и negative() предшествовать test() или могут следовать после него? Проверьте вашу гипотезу, поменяв объявления функций местами. Попробуйте объяснить результат.

Приложение 2

АНКЕТА

Ответьте на представленные ниже вопросы:

№ п/п	Показатели заинтересованности к изучению программирования	Оценка (1 – нет, 5 – да)				
		1	2	3	4	5
1	Много ли в Вашем окружении (родители, братья, сестры, друзья и т.д.) людей, профессия которых связана с программированием?					
2	Насколько интересна Вам работа в IT-сфере?					
3	Интересно ли Вам изучать языки программирования?					
4	Как часто Вы сталкиваетесь с программированием в жизни?					
5	Считаете ли Вы, что программирование – это легко?					
6	Умеете ли Вы чётко и последовательно выстраивать свои действия при решении каких-либо задач?					
7	Хотели бы Вы получать большую зарплату на должности ведущего программиста?					
8	Насколько Вы готовы углубляться в изучение программирования?					
9	Посоветуете ли Вы обучаться программированию своим друзьям?					
10	Хотите ли Вы создавать что-то новое с помощью программирования?					
11	Считаете ли Вы, что программирование может пригодиться в Вашей жизни?					
12	Насколько легким Вам кажется программирование на Python?					
13	Планируете ли Вы дальше обучаться программированию?					

Приложение 3

Оценочные средства для поверки знаний учащихся курса

Итоговое тестирование

Тест по языку программирования Python для учеников 8 класса:

1. Язык программирования Python подходит для разработки:

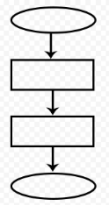
a. Компьютерных и мобильных приложений

b. Аналитика и машинное обучение

c. Игр

d. Ничего из этого.

2. Назовите тип алгоритма:



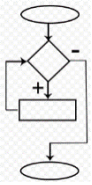
a. Разветвляющийся

b. Линейный

c. Циклический

d. Смешанный

3. Назовите тип алгоритма:



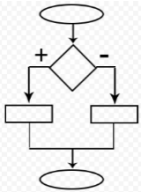
a. Линейный

b. Разветвляющийся

c. Циклический

d. Смешанный

4. Назовите тип алгоритма:



a. Разветвляющийся
b. Линейный
c. Смешанный
d. Циклический

5. Что хранит в себе переменная?

a. Имя
b. Значение
c. Тип
d. Длину своего значения

6. Что обозначает тип данных int?

a. Целочисленное
b. Вещественное
c. Строковое
d. Булевое

7. Выберите правильную запись оператора присваивания:

a. 10 = x
b. y = 7,8
c. a = 5
d. a == b + x

8. Укажите оператор ввода:

a. input()

b. print()

c. int()

d. random()

9. Сколько возможных значений у переменной типа bool?

a. 2

b. 4

c. 10

d. Сколько угодно

10. Какой оператор здесь используется?

if n < 100:

b = n + a

a. Условный оператор

b. Оператор присваивания

c. Оператор сложения

d. Оператор умножения

Ключ к тесту:									
1	**2**	**3**	**4**	**5**	**6**	**7**	**8**	**9**	**10**
a,b,c	b	c	a	b	a	c	a	a	a,b,c