МИНИСТЕРСТВО ПРОСВЕЩЕНИЯ РОССИЙСКОЙ ФЕДЕРАЦИИ
федеральное государственное бюджетное образовательное учреждение высшего образования
КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ им. В.П.Астафьева
(КГПУ им. В.П. Астафьева)

Институт математики, физики и информатики
Кафедра информатики и информационных технологий в образовании

КИСЛОВА ДАРЬЯ ВЛАДИМИРОВНА

МАГИСТЕРСКАЯ ДИССЕРТАЦИЯ

МЕТОДИКА ОБУЧЕНИЯ ПРОГРАММИРОВАНИЮ СТУДЕНТОВ СРЕДНЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ С ПОМОЩЬЮ СИСТЕМЫ ИСПОЛНИТЕЛЕЙ ВОЗРАСТАЮЩЕЙ СЛОЖНОСТИ

Направление подготовки 44.04.01 Педагогическое образование
Направленность (профиль) образовательной программы
Информационные и суперкомпьютерные технологии в математическом образовании

ДОПУСКАЮ К ЗАЩИТЕ:

Заведующий кафедрой
проф., д-р пед. наук, канд. физ.-мат. наук
Пак Н.И.
_____

Руководитель
магистерской программы
проф., д-р пед. наук, канд. физ.-мат. наук
Майер В.Р.
_____

Научный руководитель
канд. физ.-мат. наук,
Романов Д.В. Романов
_____

Дата защиты
_____

Обучающийся
Кислова Д.В. KD
_____

Оценка_____

Красноярск 2020

Реферат

магистерской диссертации

Кисловой Д.В.

по теме «Методика обучения программированию студентов среднего профессионального образования с помощью системы исполнителей возрастающей сложности»

Постановка проблемы: Проблема связана с высоким порогом первоначального входа в сам процесс программирования и, зачастую, малым количеством выделенных часов на изучение темы – влияние подобных негативных факторов на учебный процесс приводит к тому, что остаточные знания учащихся ничтожны и бесполезны для дальнейшего использования. К самостоятельной работе ученик также не способен, поскольку культура самостоятельного написания программы у него не успевает выработаться, а без неё методические рекомендации в любой форме бесполезны.

При использовании традиционных методик преподавания языков программирования, алгоритмическое мышление является финальной ступенью освоения курса, и, как таковая, она последней формируется и первой страдает от любых дефицитов и недостатков образовательного процесса, включая недостаточное количество часов, пропуски, дефициты необходимых УУД, отсутствие мотивации и интереса, из-за чего курс «Информатики» часто не выполняет эту ключевую задачу — развитие алгоритмического мышления ученика.

Цель исследования: разработать методику обучения программированию студентов СПО, позволяющую сформировать долгосрочные остаточные компетенции, достаточные для практического применения и дальнейшего развития при обучении в учреждениях более высокой ступени.

Задачи исследования:

1. Проанализировать методику преподавания программирования в школе и СПО.

2. Выполнить обзор литературы по теме исследования и смежным темам.
3. Проанализировать проекты, основанные на мотивации обучающегося игровыми, групповыми, творческими, соревновательными и социальными методиками и методами продвижения.
4. Разработать среду с визуализацией исполнителя, процесса исполнения, и результата исполнения алгоритма.
5. Разработать систему автоматической проверки задания и визуализацией ошибки.
6. Разработать систему проверки соблюдения дополнительных ограничений и визуализации их нарушения.
7. Спроектировать систему уроков.
8. Подготовить протокол анализа результативности обучения по разработанной методике.

Результаты исследования: Разработана методика обучения языку программирования Паскаль студентов СПО, нацеленная на формирование долгосрочных остаточных знаний, необходимых и достаточные для практического применения в профессиональной деятельности и дальнейшего развития при обучении в учреждениях более высокого уровня.

«__28__»___декабря____2020 г. 

(подпись / Ф.И.О.)

Summary of master's thesis of

Kislova D.V.

on the topic: « Methods of teaching programming to students of secondary vocational education using a system of performers of increasing complexity»

Statement of the problem: The problem is associated with a high threshold of initial entry into the programming process itself and, often, a small number of allocated hours for studying a topic - the influence of such negative factors on the educational process leads to the fact that the residual knowledge of students is negligible and useless for further use. The student is also not capable of independent work, since he does not have time to develop a culture of independent writing of a program, and without it, methodological recommendations in any form are useless.

When using traditional methods of teaching programming languages, algorithmic thinking is the final stage of mastering the course, and, as such, it is the last to be formed and the first to suffer from any deficiencies and shortcomings of the educational process, including insufficient hours, gaps, deficiencies of the necessary ELC, lack of motivation and interest. , which is why the course "Informatics" often does not fulfill this key task - the development of the student's algorithmic thinking.

Purpose of research: to develop a methodology for teaching programming to secondary vocational education students, which allows them to form long-term residual competencies sufficient for practical application and further development when studying in institutions of a higher level.

Research problems:

1. Analyze the methodology of teaching programming in school and open source.

2. Perform a literature review on the research topic and related topics.

3. Analyze projects based on the student's motivation by playing, group, creative, competitive and social methods and promotion methods.

4. Develop an environment with visualization of the performer, the execution process, and the result of the algorithm execution.

5. Develop a system for automatic verification of the task and visualization of errors.

6. Develop a system for checking compliance with additional restrictions and visualizing their violation.

7. Design a lesson system.

8. Prepare a protocol for analyzing the effectiveness of training according to the developed methodology.

Results of the research: A methodology for teaching the Pascal programming language to students of secondary vocational education has been developed, aimed at the formation of long-term residual knowledge necessary and sufficient for practical application in professional activities and further development when studying in higher-level institutions.

«_*20*_» __*december*___2020 yr.

_____

(Signature / FML)

ОГЛАВЛЕНИЕ

ВВЕДЕНИЕ

Внедрение цифровой техники в различных отраслях производства происходит довольно широко, в том числе и в сельском хозяйстве. Цифровизация затронула все отрасли экономики. В сельском хозяйстве трансформация рынка труда вызвана приходом современной техники и технологий и сопровождается растущим барьером между состоянием кадрового резерва малых городов и ПГТ, уровнем их образования и необходимыми для работы в сельскохозяйственной отрасли компетенциями [1].

Тенденции развития сельского хозяйства направлены на освоение и развитие систем «точного земледелия» [2], включающих в себя комплекс высокотехнологичных систем сельскохозяйственного менеджмента, в который входят:

- технологии глобального позиционирования (GPS) [3];
- географические информационные системы (GIS) [4];
- технологии оценки урожайности (Yield Monitor Technologies) [2];
- технологию переменного нормирования (Variable Rate Technology) [2];
- технологии дистанционного зондирования земли (ДЗЗ) [5].

Благодаря развитию данного комплекса стало возможным внесение удобрений и орошение посевов дифференцированным способом, своевременное реагирование на угнетение части полей, возможность более точно вносить дозированный объем прикормок для увеличения урожайности. Этот комплекс позволяет не только увеличить урожайность, но и снизить расходы, т.к. внесение необходимых удобрений идет не по всему полю, а в определенный сектор, в связи, с чем и уменьшаются расходы удобрений и топлива; например, при использовании GPS-навигации строится наиболее оптимальный маршрут следования трактора, а также осуществляется его непосредственное позиционирование на поле в реальном времени.

В настоящее время и сама сельскохозяйственная техника не стоит на месте – наряду с тракторами и комбайнами нашли свое применение и беспилотные летательные аппараты (БЛА) [6], которые также выполняют ряд работ в рамках комплекса «точного земледелия». С помощью БЛА также можно точечно вносить удобрение, и в этом случае расход ГСМ стремится к нулю, т.к. БЛА работают на аккумуляторных батареях, и нет необходимости заправлять топливом более крупную технику, дополнительно, исключаются наезды на посевы.

Функции БЛА в сельском хозяйстве гораздо шире, на них также возлагается картирование полей, а именно:

- составление точной карты поля на местности, что позволяет уточнить засеваемые площади;
- составление карт урожайности, которые дают агроному уникальные аналитические данные и характеристики поля.

Поскольку традиционные образовательные институты не успевают за стремительно меняющимся и усложняющимся рынком труда, крупные корпорации начинают готовить кадры самостоятельно, конкурируя друг с другом, как, например, компании «Яндекс. Лицей» [7] и «IT ШКОЛА SAMSUNG» [8]. Не исключение и крупные игроки рынка сельхозтехники.

Следить за последними тенденциями в развитии техники и внедрять в образовательный процесс в СПО позволяет поддержка от ведущих компаний России и мира в данной отрасли, таких как компании ООО «Геоскан», «Тимбермаш Байкал», «Группа Ростсельмаш», ООО «Мировая Техника», «Редстар оборудование», «Агромастер Красноярск», «Назаровский агроснаб», ОАО «Галактика», «Агролидер», «Канская агросельхозтехника», ООО «Лемкен-Рус», «Бизон». С их помощью в образовательных организациях оснащаются специализированные классы, позволяющие студентом изучить последние тенденции в развитии техники, а преподавателям и мастерам производственного обучения пройти стажировку от представителей компании.

Например, в настоящий момент в КГБПОУ «Уярском сельскохозяйственном техникуме» организованы специализированные классы John Deere от «Тимбермаш Байкал», Ростсельмаш от «Группа Ростсельмаш», Lemken от «Бизон», активно используются БЛА GEOSKAN от компании ООО «Геоскан». Тесное партнерство с ведущими компаниями производящими современную сельскохозяйственную технику, позволяет улучшить обучение и сделать выпускника конкурентоспособным на рынке труда, но не берёт на себя формирование фундаментальных компетенций.

Не только корпорации, но и государство создают альтернативные способы поиска и развития кадрового резерва. Один из самых крупных проектов АСИ – олимпиада НТИ – прямо нацелен на подготовку и поиск будущих технологов, инженеров, программистов независимо от школы. Помимо ежегодных акций студентам и школьникам доступно участие в олимпиадах, одной из таких является Всероссийская инженерная олимпиада НТИ для школьников. «Олимпиада НТИ» – это перспективная система командных инженерных состязаний школьников, дающих привилегии при поступлении в университеты России [9].

Олимпиада осуществляет свою работу по различным направлениям: в частности раздел «Техника» состоит из тем «Автономные транспортные системы», «Системы связи и дистанционного зондирования земли», «Беспилотные и авиационные системы» и др. Все вышеперечисленные темы раздела выполняют частные задачи, отведенные и реализуемые «точным земледелием» в сельском хозяйстве [9].

К подобным акциям можно отнести ежегодную всероссийскую акцию «Час кода», направленную на популяризацию изучения информатики и программирования среди молодежи. В рамках этой акции школьникам рассказывают о профессии программиста и новых технологиях, а также предлагают самим попробовать заняться программированием. Все занятия проводятся в легкой и интересной форме, так что принять участие в ней могут даже самые младшие классы. Сами задания подготовлены партнерами акции – ведущими компаниями

российской IT-отрасли: «Лаборатория Касперского», «Майкрософт», «Кодвардс» и «Зептолаб». По окончании «Часа кода» все участники могут получить памятные сертификаты [10].

Подытоживая, можно сказать, что перед будущими выпускниками, планирующими работать в данной отрасли, стоит необходимость свободно владеть компетенциями агронома, оператора беспилотных летательных систем, оператора современной сельскохозяйственной техники и оборудования, механика, IT-специалиста с навыками применения программного обеспечения (в том числе специального) и программирования.

К нему будут предъявляться высокие требования, проверяемые конкурсами профессионального мастерства, сдачей экзаменов на независимую оценку квалификации или в виде демонстрационного экзамена, а также требования, предъявляемые работодателями. Он должен:

- пользоваться инструментами и данными платформ, программ, позволяющих провести точный анализ ландшафта, химического состава почв сельскохозяйственных угодий;
- создавать карты индексов растительности (NDVI);
- анализировать рельеф местности для определения стратегии использования почв и планирования почвозащитных мероприятий;
- работать с гео-информационными системами и данными дистанционного зонирования земли, с агрономическими метеоданными, включая способность анализировать данные метеопрогноза для принятия агрономических решений;
- работать с компьютерными системами поддержки принятия решений (СППР);
- организовывать техническое планирование работ для тракторов и сельскохозяйственных машин;

- создавать карты-предписания для проведения агротехнических работ на тракторе, передавать карты-предписания на трактор и другую сельскохозяйственную технику;
- проводить фактический анализ выполнения работ на технике;
- владеть специальными инструментами для диагностики состояния сельскохозяйственной техники.

В связи с этим у работодателя возникает ряд требований к соискателю, таких как: свободное владение современными технологиями, навыки настройки, отладки и управления техническими устройствами и программным обеспечением, знание современных тенденций в развитии отрасли. Работодатель не готов тратить время и деньги на обучение или переобучение сотрудника при трудоустройстве, в связи с этим все вышеперечисленные требования ложатся на плечи образовательных организаций высшего и среднего звена и одной из фундаментальных дисциплин, одна из которых, «Информатика», позволяет изучить последние тенденции в развитии цифровой техники. Также отсутствие сформированных ключевых базовых компетенций и УУД, низкий IQ делают такое переобучение практически невозможным [11].

Для успешного усвоения и отработки навыков при работе с современными комплексами в сельском хозяйстве у выпускника должны быть развиты навыки самостоятельной работы, планирования работы и принятия решений, развито алгоритмическое и вычислительное мышление, т.к. работа современных комплексов связана с работой на компьютере, через который происходит управление цифровыми устройствами, например: настройка и отладка полетного задания для БЛА, отработка навыков GPS навигации, картирования полей и т.д.

Развитие алгоритмического мышления является фундаментальной задачей при освоении дисциплины «Информатика», одна из основных тем посвящена становлению навыков алгоритмического мышления – и в тоже время является одной из проблемных тем дисциплины.

При обучении студентов СПО алгоритмизации и программированию есть ряд сильных системных противоречий:

- отсутствие ряда формируемых в старшей школе компетенций – и требование к развитому алгоритмическому мышлению, необходимому для оперирования и программирования современной сельскохозяйственной техники;
- недостаточное количество отводимых на само программирование часов и необходимость длительной практической работы для формирования высокоуровневых компетенций и освоения абстрактных понятий;
- абстрактный характер дисциплины и предельно практическая направленность самих образовательных программ и интересов выпускников;
- отток наиболее талантливой и способной части кадрового резерва и инновационное давление большого бизнеса, который входит на новые и существующие рынки с арсеналом из наиболее современных и конкурентоспособных технологий и бизнес-подходов;
- ускоряющийся темп развития техники, методик построения, и общая динамичность бизнес-процессов, – и постоянно растущая нагрузка на образовательные учреждения, приводящая к дефициту преподавателей, конкурентных на рынке труда будущих выпускников;
- короткий цикл обновления техники и отсутствию у выпускника фундаментальной образовательной базы в области ИТ-технологий, позволяющей проходить переподготовку «малой кровью»;
- растущие требования к квалификации преподавателя, и его обучение, проходившее на старой технической базе и методической литературе, сопровождаемое отсутствием времени

для качественного повышения квалификации (дефицит квалифицированных кадров есть на всех рынках образования);

– частое отсутствие практического знания о современных технологиях в сельском хозяйстве у преподавателя СПО.

В обзоре литературы и первой главе работы эти противоречия проанализированы и раскрыты с нескольких позиций:

– проанализировано понятие алгоритмического мышления как компетенции, позволяющей оператору выстраивать ментальную модель сложного устройства и решать с её помощью поставленные задачи, выполняя необходимую декомпозицию задач на подзадачи, доступные решению имеющимися средствами;
– особенности формирования вычислительного и алгоритмического мышления студентов СПО;
– проанализированы понятие программы и деятельность ученика и педагога в ходе освоения дисциплины «Информатика» с целью выделить ключевые аспекты, недоработки в которых критически сказываются на образовательных результатах;
– понятие вычислительного мышления как навыка XXI-го века, века мощных аналитических и вычислительных инструментов, где от специалиста требуется уже не умение решать сложные задачи руками, а формулировать проблемы таким образом, чтобы можно было использовать компьютер и другие инструменты для их решения.

Цель исследования: разработка методики обучения программированию студентов СПО и формирования долгосрочных остаточных компетенций, достаточных для практического применения и дальнейшего развития при обучении в учреждениях более высокой ступени.

Проблема связана с высоким порогом первоначального входа в сам процесс программирования и, зачастую, малым количеством выделенных

часов на изучение темы – влияние подобных негативных факторов на учебный процесс приводит к тому, что остаточные знания учащихся ничтожны и бесполезны для дальнейшего использования. К самостоятельной работе ученик также не способен, поскольку культура самостоятельного написания программы у него не успевает выработаться, а без неё методические рекомендации в любой форме бесполезны.

Другими словами, при использовании традиционных методик преподавания языков программирования, алгоритмическое мышление является финальной ступенью освоения курса, и, как таковая, она последней формируется и первой страдает от любых дефицитов и недостатков образовательного процесса, включая недостаточное количество часов, пропуски, дефициты необходимых УУД, отсутствие мотивации и интереса, из-за чего курс «Информатики» часто не выполняет эту ключевую задачу — развитие алгоритмического мышления ученика [12].

Это обусловливает **актуальность** настоящей работы.

Объектом исследования является процесс обучения языку программирования студентов 1 курсов СПО.

Предметом исследования является формирование навыка алгоритмического и вычислительного мышления.

Гипотезой работы является: Достичь удовлетворительной результативности обучения студентов СПО программированию можно, перестроив форму и характер работы. Старая форма – это освоение языка программирования Паскаль в ходе курса «Информатика», включающее:

- теоретические блоки;
- демонстрации;
- воспроизведение на практике;
- самостоятельное решение серии искусственно упрощенных задач;
- тестирование и проверки;
- переход к более сложным задачам;

– использование языка для решения задач в других областях науки и практики.

Этот подход требует достаточно большого количества аудиторной работы под контролем педагога для формирования УУД и компетенций на требуемом уровне, и не реализуется в настоящее время во многих СПО. Новая форма работы предполагает предварительное развитие алгоритмического мышления и освоение ключевых понятий программирования (условный оператор, цикл, функция) во *внеурочное* время – путём прохождения системы квестов, выполняемых визуальным автоматическим исполнителем. При переходе к работе в классе, педагог заменяет исполнителя, визуально и функционально тождественного предыдущим, только программируемого на языке Паскаль, что позволяет закрепить уже полученные навыки и изучить недостающие элементы курса (понятие переменной, типа данных и т.д.) сверху-вниз, с опорой на уже освоенную базу задач и решений. Аналогичный подход, с опорой на визуальную обратную связь, использовался при разработке языка КуМир [13]; в нашем случае мы заменили исполнителя на игру с низким порогом входа и перешли, не меняя внешней формы, на использование языка Паскаль, уже учитывая возраст учеников.

При проектировании системы заданий, на решение будут накладываться ограничения, выполнимые только при использовании и понимании основных алгоритмических конструкций, являющихся фундаментом алгоритмического и вычислительного мышления [14]. В игре также предусмотрена обратная связь с учеником и сохранение для педагога выполненного задания.

Игровая форма делает саму постановку задачи обучения неявной, снимая давление на ученика и смещая роль учителя с преподавателя на комбинацию ролей ментора и тьютора (и фасилитатора при групповой работе в классе).

ГЛАВА 1. АЛГОРИТМИЧЕСКОЕ МЫШЛЕНИЕ И МЕТОДЫ ЕГО ФОРМИРОВАНИЯ

Информатика как обязательный школьный предмет берет своё начало с 1984 г. под руководством А.П. Ершова. С тех пор техника шагнула далеко вперед. На данный момент значение информатики гораздо шире, роль её заключается не только в изучении современных программ, позволяющих упростить и ускорить какие-либо расчеты, а в целом изучении цифровизации отрасли – компьютерные технологии становятся средством реализации технических средств производства. Оператор цифровой техники уже не привязан к стационарному рабочему месту, он мобилен и способен решать производственные задачи удалённо.

Простая компьютерная грамотность переросла в «знание "информационных технологий"», куда включают изучение программных средств информатизации, владение системным, сервисным и прикладным программным обеспечением. Эти компетенции даются в курсе «Информатики». Нас этот курс интересует с точки зрения развития алгоритмического и вычислительного мышления. Под «вычислительным мышлением» в научной и методической литературе, посвящённой проблемам обучения информатике и программированию, понимают мыслительные процессы, задействованные в решении проблемы, которая может быть выражена в виде последовательности шагов (алгоритма) и может быть решена с помощью компьютера [16].

Вычислительное мышление как «процесс решения проблем, включает (но не ограничивается ими) следующие характеристики:

- формирование проблем таким образом, чтобы можно было использовать компьютер и другие инструменты для их решения;
- анализ и логическое упорядочивание данных, представление их с помощью абстракций;
- автоматизация решений с помощью алгоритмов (последовательности упорядоченных шагов);

– выявление, анализ и внедрение возможных решений с целью достижения наиболее эффективного и действенного сочетания алгоритмов и ресурсов;
– обобщение и перевод этого процесса решения проблем на широкий спектр задач».

Вычислительное мышление включает в себя широкий спектр интеллектуальных инструментов и теорий из информатики, которые помогают решать проблемы, проектировать системы, понимать поведение человека и использовать компьютеры для автоматизации широкого спектра задач [16].

Под «алгоритмическим мышлением» понимают специфический тип мышления, предполагающий умение создавать алгоритм решения различных задач. Алгоритмическое мышление является важной составляющей интеллектуального развития человека.

Алгоритмическое мышление включает в себя ряд особенностей, свойственных логическому мышлению, однако требует и некоторых дополнительных качеств. Основным из них считается умение находить последовательность действий, необходимых для решения поставленной задачи, а также выделение в общей задаче ряда более простых подзадач, решение которых приведет к решению исходной задачи.

Изучение вопросов развития мышления в процессе проблемного обучения на уроке информатики, в частности обучения различным языкам программирования, позволило сделать вывод о том, что методика проведения занятий, используемые средства, специфика решаемых проблемных задач, цели и качество получаемого результата должны быть направлены, в основном, на развитие логического и алгоритмического стиля мышления.

«Алгоритмический стиль мышления - это система мыслительных способов, действий, приёмов, которые направлены на решение как теоретических, так и практических задач, и результатом которых являются алгоритмы, как специфические продукты человеческой деятельности».

Алгоритмическое мышление помогает формировать следующие умения и навыки:

- планирование структуры действий, необходимых для достижения определенной цели, используя фиксированный набор ресурсов;
- создание информационной структуры для описания объектов и средств;
- организация поиска информации, необходимой для решения проблемы;
- правильная, четкая и недвусмысленная формулировка идеи в понятной форме и правильное принятие текстового сообщения;
- своевременное использование компьютера при решении задач из любой области;
- формирование навыков анализа и структурирования информации [15].

Из определений ясно, почему этот тип мышления считается важной компетенцией: прежде чем применять вычислительные инструменты для решения какой-либо задачи или проблемы, необходимо понять саму проблему и способы ее решения. Речь идет о переформировании проблемы и структурировании ее таким образом, чтобы можно было решить ее с помощью компьютера. Алгоритмическое мышление позволяет это выполнить, и рассматривается как важная компетенция квалифицированного специалиста и выпускника. Формирование алгоритмического и вычислительного мышления является одной из основных проблем современного образования [16].

Полностью осознавая важность развития алгоритмического и вычислительного мышления, многие специалисты исследовали вопросы их формирования в школе и ВУЗе. Стоит отметить, что развить алгоритмического мышления в рамках одной дисциплины очень тяжело, и эта фундаментальная задача стоит не только перед СПО, но и перед школами. Для развития такого рода мышления важен системный подход, и

чем раньше с учащимся начнется такая работа, тем успешней будут результаты, и безболезненнее будет переход к применению данного типа мышления в какой-либо другой деятельности.

Ниже перечислены результаты, полученные рядом авторов в этом направлении.

Методика Н.Н. Еремеевой: Существует методика, позволяющая начать развитие алгоритмического мышления еще с начальной школы при работе класса в малых группах, в данной методике описываются правила взаимодействия детей в группе при решении проблемных ситуаций способствующих формированию алгоритмического мышления [17].

Методика И.Р. Дединского: Характеризуется наличием системного подхода и принципом «Учить только хорошему»: на первом же занятии дети узнают, как правильно пользоваться пробелами и отступами, и почему важны пустые строки, разбивающие программу на логические фрагменты. С первых же занятий вводится понятие качества имен, и от детей требуется использовать понятные имена для переменных и функций [18].

Её отличает аналитический подход к довузовскому преподаванию программирования: подход, ориентированный на проектную работу, сильно увлекает многих учеников и дает не только высокие проектные результаты, но и высокие олимпиадные. Главная учебная задача, и не только в сфере ИТ, – научить студента действовать грамотно и самостоятельно [18].

Методика Т.П. Пушкаревой: визуальная методик обучения программированию разделена на три этапа: 1. Визуализация понятий программирования и алгоритмических конструкция; 2. Построение алгоритмов и различные способы записи; 3. Применение методов для длительного запоминания понятий и конструкций [19].

Методика Е.Ф. Родыгина: методические рекомендации обучения программированию в школе, изучение основ программирования разделены на этапы: 1. Выбор языка; 2. Организация обучения (лекции, лабораторные и практике); 3. Технология обучения. Изучать языки

программирования начинают с 8 по 11 классы, с постепенным усложнением тем [20].

Есть также точка зрения, что начинать изучение темы «Основы алгоритмизации и программирование» целесообразно со знакомства с исполнителями с обратной связью. Такое обучение обеспечит подготовку к последующему изучению языков программирования высокого уровня в наглядной форме. Рассмотрение базовых алгоритмических конструкций, в применении с программируемым устройством, позволит сформировать навыки их использования при решении более сложных практических задач. В сравнении с исполнителями виртуальными, реальные исполнители послужат фундаментом для формирования умения проанализировать задачу и формализовать ее условие применительно к условиям реальной обстановки [21].

Наряду с мобильными приложениями существуют обучающие программы-тренажёры, которые также призваны в дружелюбной форме познакомить учащегося с языками программирования.

К таким программам-тренажерам и исполнителям можно отнести:

- Виртуальные исполнители алгоритмов ЛОГО-Черепашка, Комплект Учебных Миров, Паркетчик, Scratch.
- Реальные исполнители с обратной связью FISCHERTECHNIK, Robotis Bioloid, LEGO MINDSTORMS, EV3 [21].
- «Школьница». Входными языками системы являлись языки Робик (8 - 12 лет) и Рапира, дополненные встроенным межъязыковым модулем — графической системой Шпага [22].
- Веб программы-тренажеры contest-er и acmp.ru, acm.timus.ru, codeforces.ru, informatics.msk.ru, olymp.isu.ru [23].
- Использование web-визуализаторов [24].

Но т.к. студенты приходят в техникум на базе 9 класса с разных школ и с абсолютно разным уровнем подготовки, то перед нами стоит задача развивать алгоритмическое мышление, начиная только с занятий информатики на 1 курсе.

Из-за постоянного наполнения программ СПО новыми предметами, количество часов на изучение дисциплины «Информатика», как правило, сокращается, и на данный момент для изучение всей дисциплины отводится порядка 150 аудиторных часов, а на раздел «Информация и информационные процессы» - 42 аудиторных часов, и из них лишь 14 часов непосредственно на тему «Основы алгоритмизации и программирование», традиционно в дисциплину входит изучение следующих тем:

- алгоритмы и способы их описания;
- построение линейных алгоритмов;
- построение алгоритмов с использованием конструкций проверки условий;
- построение алгоритмов с использованием циклических конструкций;
- введение в язык программирования;
- среда программирования;
- синтаксис программы;
- семантика программы;
- программная реализация алгоритма;
- тестирование программы.

В этом случае перед преподавателем стоит непростая задача, научить студента мыслить структурно и выделять в поставленной задачи шаги необходимые для её решения с целью развития алгоритмического мышления. Эта задача и раньше выполнялась с трудом, а в столько малое количество часов становится невозможной. В помощь преподаватель может задействовать следующие ресурсы:

- домашнюю работу;
- самостоятельную работу;
- проектную и исследовательскую деятельность;
- кружковую деятельность.

Во главе вышеперечисленного стоит задача научить организовывать самостоятельную работу учащихся, т.к. если студент в полной мере овладеет навыками самостоятельной работы, планирования, самоконтроля, то и остальные ресурсы, такие как выполнение домашней работы, участие в проектной и исследовательской деятельности, кружковая деятельность будут выполняться эффективней.

Согласно исследовательской работе Дацур Наталье [25], при организации самостоятельной работы студента предусматривается выполнение следующих шагов:

1. Входное тестирование по определениям понятий, синтаксису и семантике конструкций языка, особенности их применения.
2. «Чтение» текста программы на языке программирования и запись его семантики на естественном языке, используя терминологию текущей темы.
3. «Диктант» по тексту на естественном языке написание фрагмента программы на языке программирования с использованием данных (констант, переменных и выражений) изучаемого типа данных: их объявление, инициализация или ввод значений, обработка одним из методов решения задач на ЭВМ, вывод.
4. «Библиотека» использование наиболее распространенных функций стандартных библиотек (встроенных функций) для работы с изучаемым типом данных во фрагменте программы на языке программирования.
5. «Понимание» анализ фрагмента программы на языке программирования, который содержит типовые ошибки использования изучаемого типа данных и конструкций языка, с целью его верификации и указания перечня обнаруженных ошибок и соответствующих действий по их устранению.
6. «Сочинение» разработка программы на языке программирования для решения задачи с использованием

изучаемого типа данных и конструкции языка в соответствии с индивидуальным заданием.

7. Заключительное контрольное тестирование по определениям понятий, синтаксису и семантике конструкции языка, особенности их применения.

Без культуры следования этим шагам самостоятельная работа снижает эффективность. Формирование алгоритмического и вычислительного мышления при использовании данных ресурсов можно также реализовать с использованием:

- игровых технологий;
- олимпиадных и конкурсных заданий;
- формирование мышления в процессе обучения теории графов [26].

Данные ресурсы отлично встраиваются в учебный процесс, подходят как для аудиторной, так и для внеклассной работы, позволяют качественно и наглядно отработать практические навыки, позволяют школьникам и студентам взглянуть на мир программирования с другой, более красочной и наглядной стороны.

В процессе преподавания перед преподавателям возникают следующие проблемы:

- выбор языка программирования;
- отбор содержания обучения;
- выбор методов, форм и средств обучения;
- реализация межпредметных и внутрипредметных связей;
- уровень подготовки студента должен соответствовать уровню подготовки к ЕГЭ [27].

Для решения задачи в такой постановке было решено использовать подход родственный новой методике BYOD [28], со сменяемым исполнителем. Это позволило снизить влияние следующих негативных факторов, затрудняющих изучение языков программирования, например, Паскаля:

- языки программирования чужды студентам и вызывают отторжение;
- наличие «иностранных» команд-операторов;
- агрессивно выскакивающие ошибки;
- объем материала, который нужно каким-то образом структурировать в голове и ориентироваться в присутствии новых терминов;
- представленные задачи нужно не только решить логически устно, но и разбить на шаги, и изобразить их в графическом виде, и потом все это переложить на какой-то язык программирования.

В результате совокупного действия этих факторов, у студентов возникает полное отторжение темы и полная «каша» в голове, не позволяющая увидеть простые связи, имеющиеся в данной теме.

В соответствие с гипотезой настоящей работы, для того, чтобы у студента не возникало боязни перед языками программирования, стоит начать изучение основ алгоритмизации с наглядных инструментов с дружелюбным интерфейсам, к таким инструментам можно отнести различные логические видеоигры, для детей разных возрастов. Примером такой игры является «Algorithm City».

«Algorithm City» позволяет изучить основы программирования и алгоритмизации, обучение происходит в виде кодирующей игры, в которой учащиеся задают последовательность действий необходимые для выполнения поставленной задачи [29]. Действия персонажа выбираются с помощью предложенных блоков и с ограничением числа шагов-операций для изучения программных концепций.

В ходе игры учащиеся получают представление о базовых концепциях кодирования, такие как последовательность команд, функции и петли. Игра имеет 50 уровней в 4 главах:

1. Учебная глава имеет 5 уровней, показывает и обучает способы игры в приложении.
2. Легкая глава имеет 15 уровней, обучает основам кодирования.

3. Нормальная глава имеет 15 уровней, обучает циклам.
4. Hard Chapter имеет 15 уровней, обучает функциям.

Использовать данное приложение можно для домашней, самостоятельной или исследовательской деятельности обучающихся. Аналогов данной игры множество, можно выбрать вариант любого уровня сложности, приближенности к языкам программирования и написания кода, для любого возраста. Все они распространяются бесплатно или условно-бесплатно, устанавливаются на мобильные телефоны, планшеты или запускаются как flash-игры для браузера.

После успешного прохождение видеоигры и усвоения основных базовых конструкций учащийся без труда выделит необходимые шаги из заданного перечня действий персонажа для решения поставленной задачи, тем самым начиная мыслить структурно. И переход, к каким-либо языкам программирования происходит плавно, т.к. учащийся уже имеет представление и понимает, что такое команда, могут задать исполнителю шаги, повторяющиеся действия организовать в циклы и т.д.

ГЛАВА 2. РЕАЛИЗАЦИЯ МЕТОДИКИ

2.1. Игра «Algorithm City»

Игра предназначена для изучения основ программирования и алгоритмов в виде кодирующей игры, в которой учащиеся задают последовательность действий необходимых для выполнения поставленной задачи. Интуитивно понятный и красочный дизайн располагает и настраивает на умственную деятельность.

Обучение происходит с помощью выбора и определения действий для персонажа (пингвин, лиса, корова, божья коровка, сердитая птица, кролик, курица и т. д.).

Учащиеся получают базовые концепции кодирования, такие как последовательность команд, функции и петли, путем управления персонажем и выполнения действий в виде сбора золотых монет и решения уровней.

Игра имеет 50 уровней в 4 главах:

1. Учебная глава имеет 5 уровней, показывает и обучает способы игры в приложении.
2. Легкая глава имеет 15 уровней, обучает основам кодирования.
3. Нормальная глава имеет 15 уровней, обучает циклам.
4. Сложная глава имеет 15 уровней, обучает функциям.

Стартовое окно игры, в котором можно настроить громкость фоновой музыки и звуков игры, выбрать персонажа, оценить приложение и приступить к прохождению игры (рис. 1). При прохождении уровня персонаж набирает монеты, за которые можно открыть нового персонажа, взять подсказку, каждый уровень оценивается от 1 до 3 звездочек.

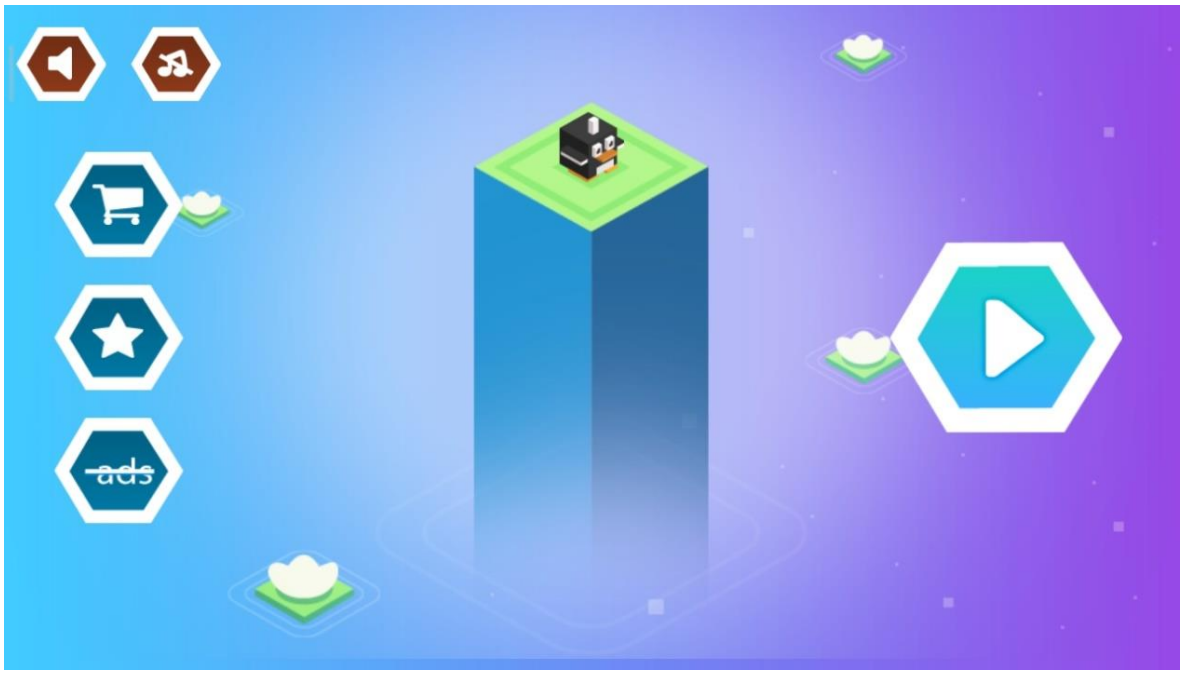


Рисунок 1 - Стартовое окно игры «Algorithm City»

Рисунок 2 - Выбор персонажа

1. **Учебная глава** имеет 5 уровней, показывает и обучает способам игры в приложении.

Обучение в этой главе начинается с самых простых действий и с минимальным набором команд: вперед, взять монетку, выбрать действие можно из блока «BLOCK AREA», при этом все необходимые шаги выполняются по указанию «руки» (рис. 3). Постепенно добавляется система команд для исполнителя, также есть ограничения по шагам, т.е. перед учащимися изначально стоит задача написать псевдокод за

определенное количество шагов, которое составляет не более 12 действий. В блоке «MAIN AREA» записываются отдельные действия – шаги персонажа. После того как все желаемые действия были отмечены в блоке «MAIN AREA», необходимо нажать на «play» и запустить выполнение программы. Если действия были выбраны с ошибкой, то уровень необходимо будет пройти заново. Стоить отметить, что каждый шаг персонажа в псевдокоде подсвечивается красным цветом, что позволяет наглядно отследить какие действия выполняет персонаж.

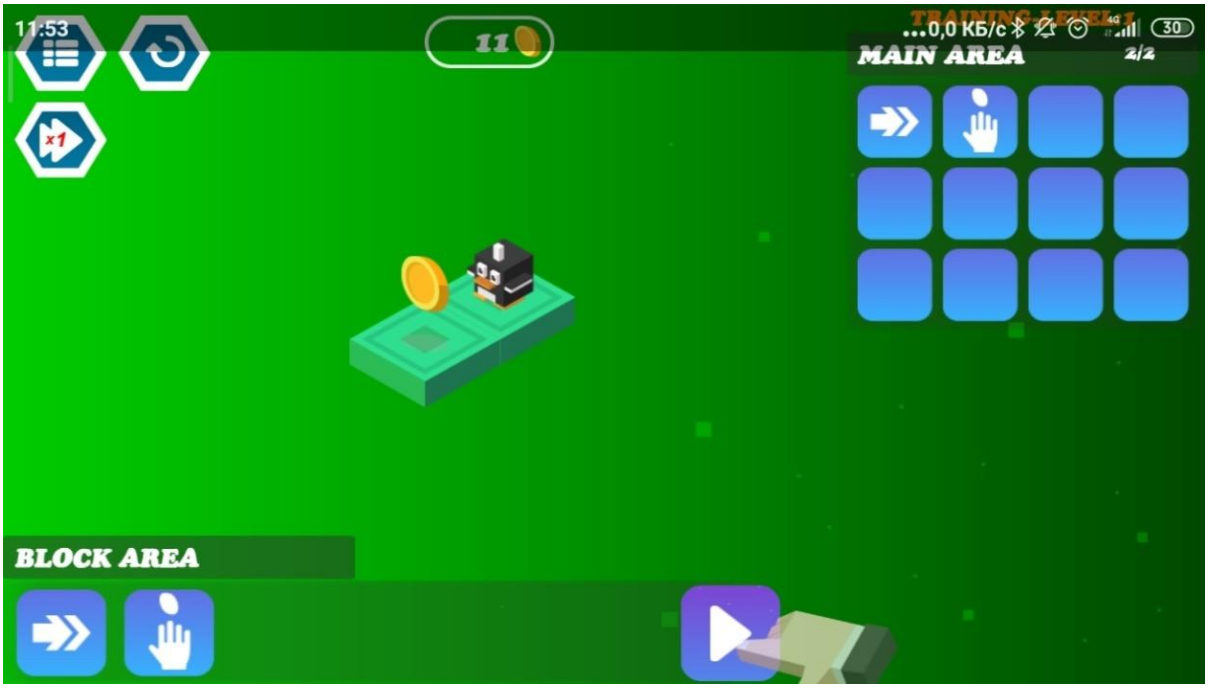


Рисунок 3 - Учебная глава

А т.к. количество действий в игре ограничено, то знакомство и использование циклов и функций происходит естественным путем, учащиеся вынуждены включать их в свои программы, т.к. они позволяют выносить из основного перечня действий какие-то повторяющиеся шаги, тем самым сокращать псевдокод и укладываться в отведенное количество действий.

Знакомство с функциями выполняются по указателю, что гарантирует их правильное применение, данный блок также имеют ограничение на количество шагов (рис. 4).

Рисунок 4 - Учебная глава. Использование функций

Совместно с функциями происходит знакомство и формирование навыков использования и выделения однотипных шагов в циклы, и умению использовать выделенные шаги многократно.

2. **Легкая глава** имеет 15 уровней, обучает основам кодирования.

В данном уровне используются только простые структуры, нет функций и циклов, действия пишутся линейно и выполняются последовательно. Отсутствуют подсказки «руки» как было в обучающей главе, но также постепенно добавляются система команд для исполнителя, есть ограничения по шагам, которое составляет не более 12 действий. Появляется новая возможность использовать подсказку, которая дается за определенное количество монет.

Уровни начинаются с самых простых и коротких программ, постепенно удлиняется маршрут, что позволяет увидеть обучающимся повторяющиеся действия, которые на данном уровне приходится прописывать «руками», далее усложняется траектория движения, добавляются повороты и прыжки. Пример данного уровня представлен на рисунке 5.

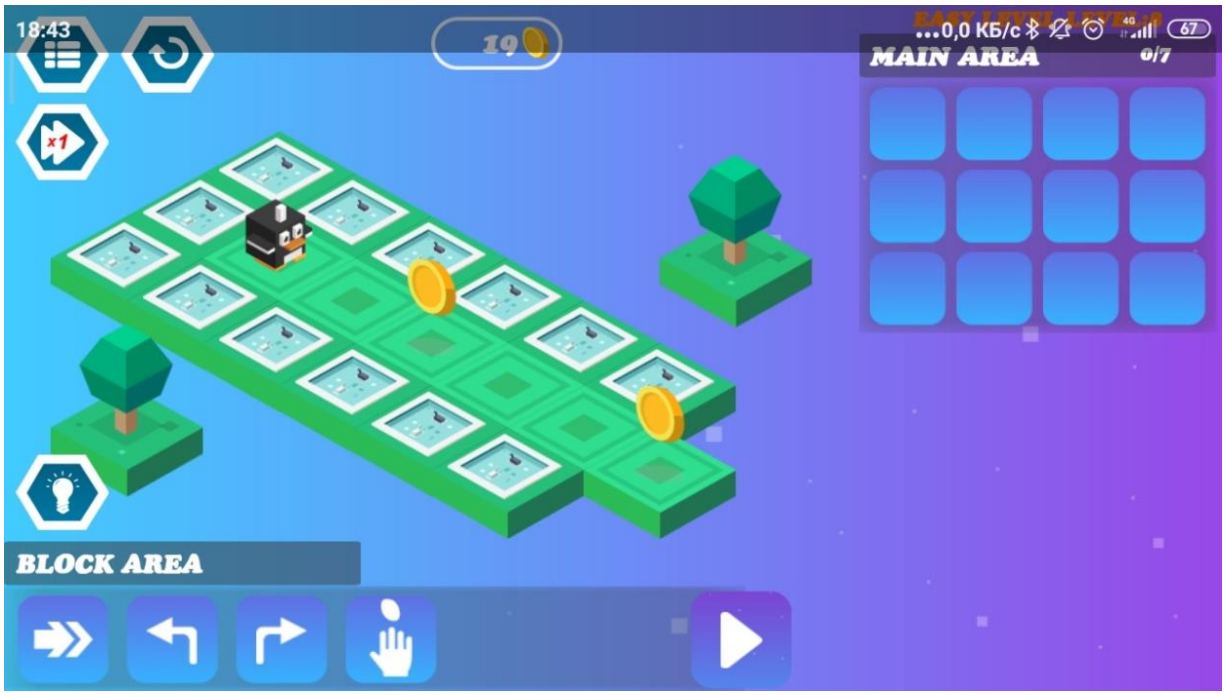


Рисунок 5 - Легкая глава

3. **Нормальная глава** имеет 15 уровней, обучает циклам с помощью функций.

В данной главе знакомство и использование циклов происходит с первых уровней, при этом усложнение заданий и добавление действий персонажу идет постепенно, так же, как в предыдущих главах игры. Основной псевдокод программы ограничен вначале одним действием, при этом открывается возможность организовать повторяющиеся шаги в отдельный блок «FUNCTION AREA f{}», который имеет ограничение на количество шагов и составляет 12 действий. Функцию при прохождении уровней можно вызывать рекурсивно внутри себя, что делает его цикличным при выполнении. Все вышеперечисленные условия ставят обучающихся в такие рамки, что позволяют сформировать понятие циклов и их использование естественным путем. Пример данного уровня представлен на рисунке 6.

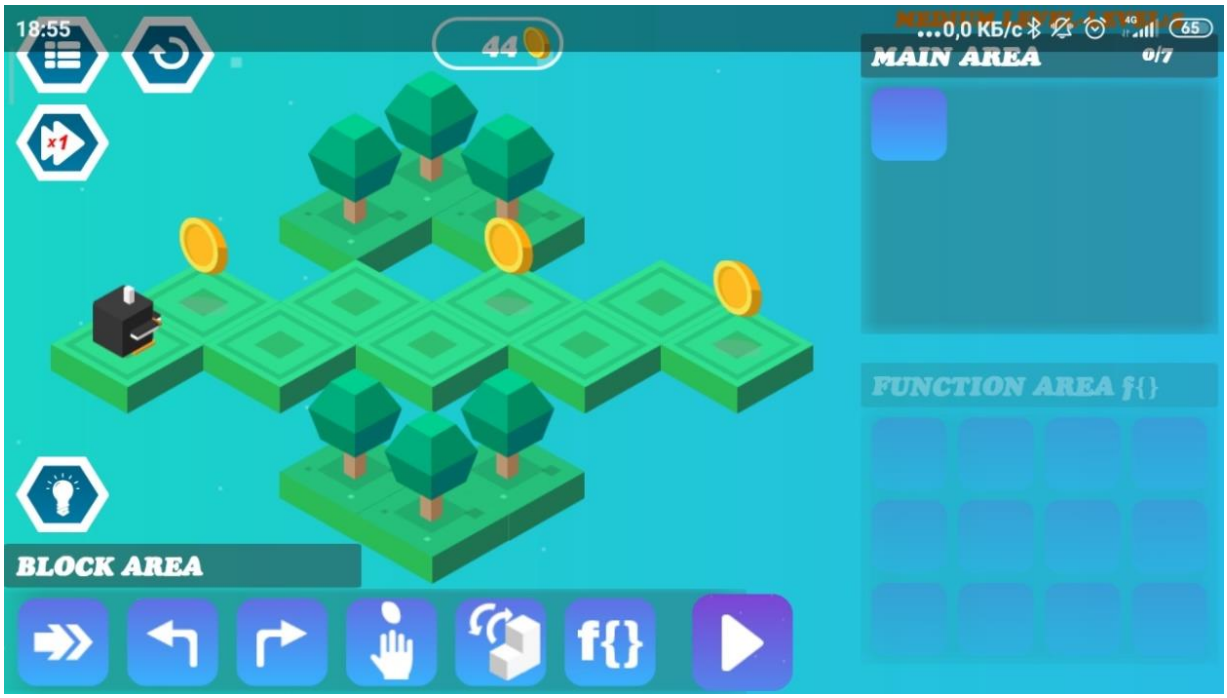


Рисунок 6 - Нормальная глава

4. **Сложная глава** имеет 15 уровней, обучает функциям.

Принципиально отличие данной главы от предыдущих в том, что блок «MAIN AREA», содержит более одного действия и обучающиеся могут вызывать функцию на исполнение несколько раз за программу, добавляя между обращениями недостающие действия. При этом блок «FUNCTION AREA f{}», перестает, работает рекурсивно, как было в предыдущей главе. Пример данного уровня представлен на рисунке 7.

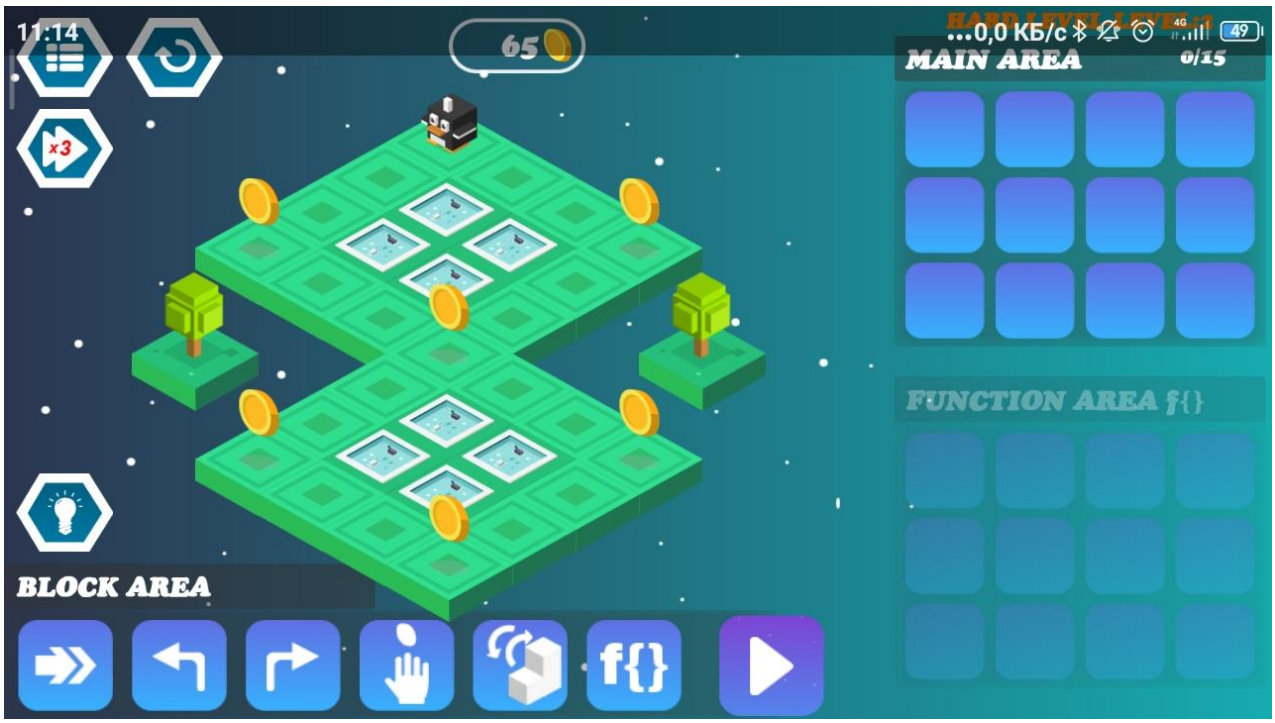


Рисунок 7 – Сложная глава

2.2. Самостоятельная работа

Для контроля формирования навыков полученных при прохождении игры и закрепления их на практике, был разработан сборник задач для самостоятельной работы, решение которых необходимо выполнить в письменном виде, что позволяет отсечь интуитивное прохождение игры, и делает обучение более осознанным.

Самостоятельная работа разделена на уровни:

- 1 уровень – Линейные алгоритмы (простые);
- 2 уровень – Линейные алгоритмы (сложные);
- 3 уровень – Циклические алгоритмы. Цикл N-раз;
- 4 уровень – Команды ветвления. Если, выбор;
- 5 уровень – Циклические алгоритмы. Цикл ПОКА.

Предварительно, перед прохождением и решением задач в сборнике есть описание персонажа, ситуацию, в которой он находится, и действия, которые может совершить. Образец описания представлен ниже.

Вы являетесь водителем трактора, которым управляете на расстоянии через систему спутниковой навигации. У вас нет руля, рычагов и педалей, все действия запрограммированы на клавиатуру вашего компьютера, т.е. команды трактору можете передавать с помощью клавиш или кодовых слов.

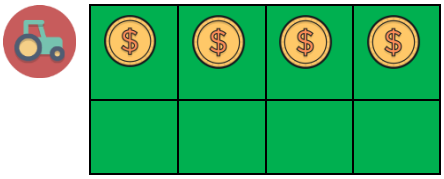
Трактору доступны действия:

Действие	Клавиша	Слово
Ехать вперед *(трактор передвигается вперед на одну клетку)*		*Вперед*
Повернуть направо *(трактор на месте поворачивает направо)*		*Направо*
Повернуть налево *(трактор на месте поворачивает налево)*		*Налево*
Вспашка *(трактор вспахивает одну клетку, на которой находится)*		*Вниз*
Взять *(трактор берет тот объект, который находится на одной клетке вместе с ним)*		*Вверх*

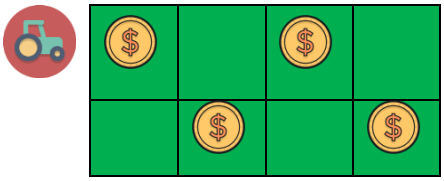
1 уровень – Линейные алгоритмы (простые). В данном уровне разработаны простые задачи, с последовательным усложнением обстановки и увеличением количеством действий необходимых совершить для решения задачи. Образец задач 1 уровня представлен ниже:

Задачи:

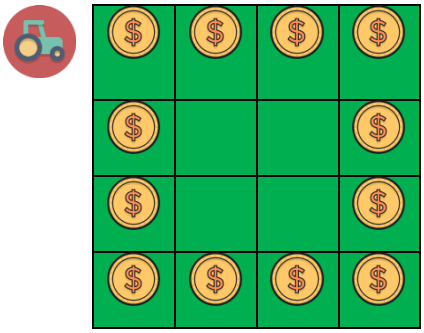
1. Перед вами поле размером 4×2, в начале поля стоит трактор, ему необходимо проехать по полю и собрать все монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



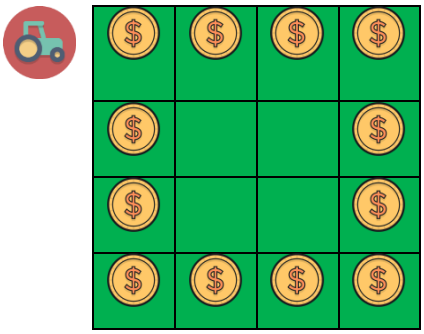
2. Перед вами поле размером 4×2, в начале поля стоит трактор, ему необходимо проехать по полю и собрать все монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



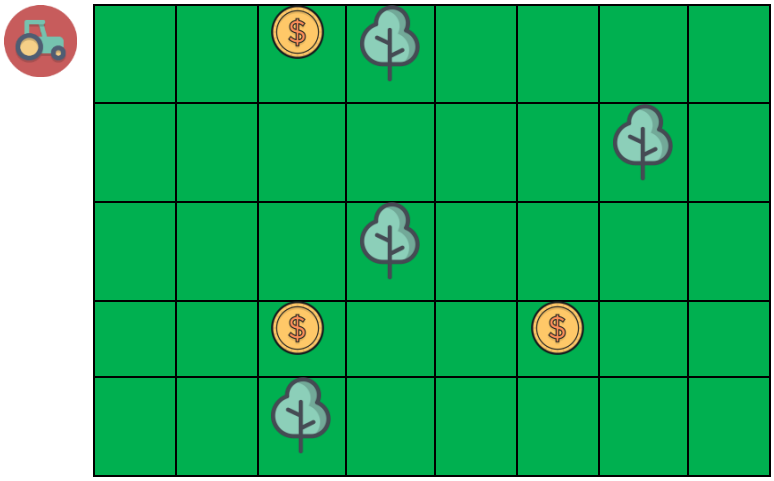
3. Перед вами поле размером 4×4, в начале поля стоит трактор, ему необходимо проехать по контуру поля и собрать все монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



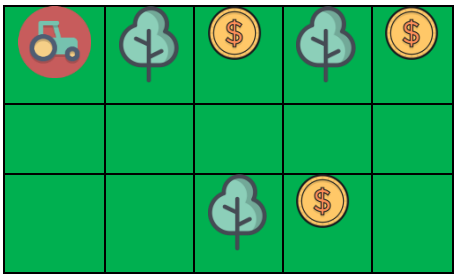
4. Перед вами поле размером 4×4, в начале поля стоит трактор, ему необходимо проехать по контуру поля и собрать все монетки и вспахать поле по контуру. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



5. Перед вами поле размером 8×5, в начале поля стоит трактор, ему необходимо проехать по полю и собрать все монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



6. Перед вами поле размером 5×3, в начале поля стоит трактор, ему необходимо проехать по полю и собрать все монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



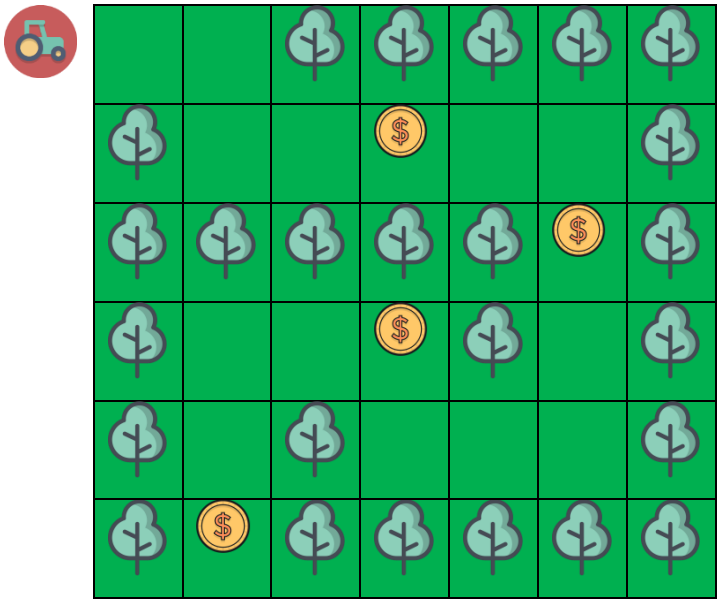
2 уровень – Линейные алгоритмы (сложные).

После того как обучающиеся освоились в написании простого набора команд при решении задач, этот же самый линейный тип алгоритма усложняется по средствам добавление действий перед персонажем (собрать монеты, вскопать участок), расширение обстановки в которой он находится. Также вводятся задачи с многократным повторением однотипных действий для того чтобы подвести обучающихся

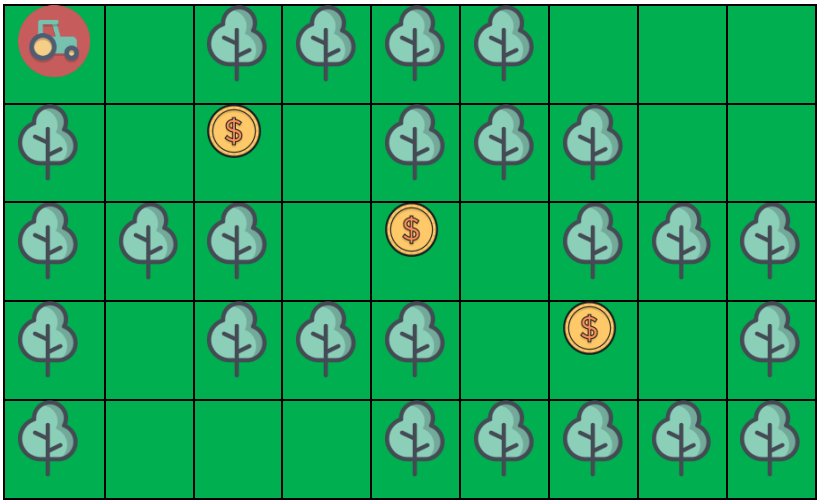
в целесообразности использования циклов при решении задач. Образец задач 2 уровня представлен ниже:

Задачи:

7. Перед вами поле размером 7×6, в начале поля стоит трактор, ему необходимо проехать по полю и собрать все монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?

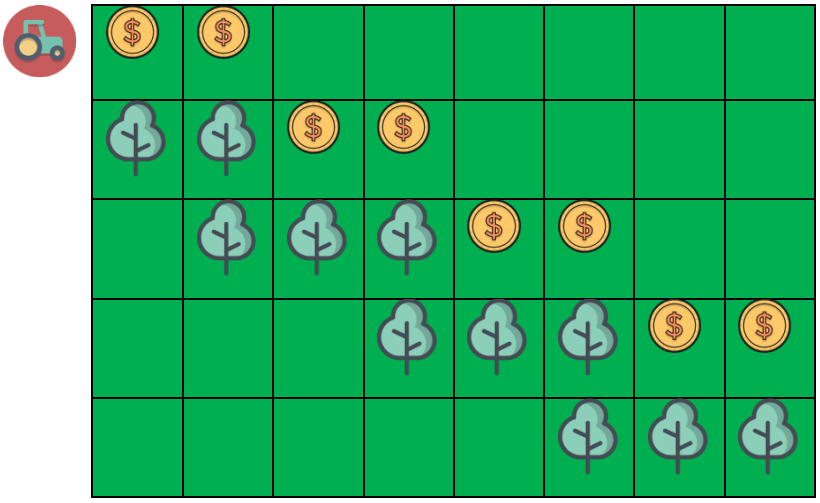


8. Перед вами поле размером 9×5, в начале поля стоит трактор, ему необходимо вспахать все поле и собрать монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?

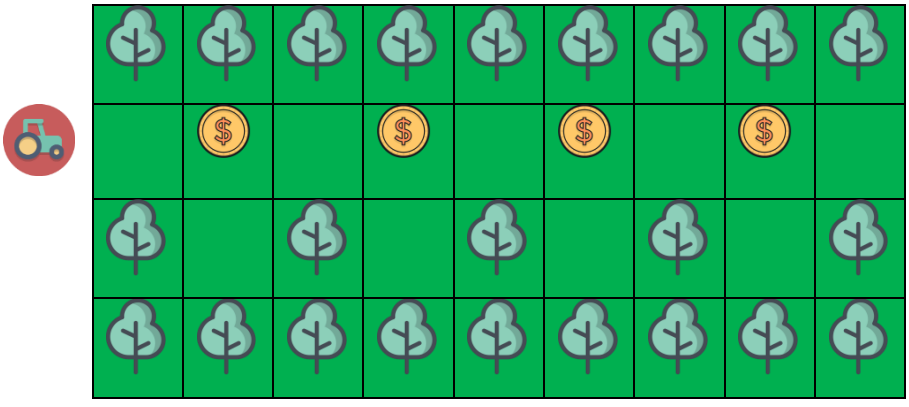


9. Перед вами поле размером 8×7, в начале поля стоит трактор, ему необходимо вспахать поле в обозначенных местах  и собрать монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?

10. Перед вами поле размером 8×5, в начале поля стоит трактор, ему необходимо собрать все монетки на поле. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



11. Перед вами поле размером 9×4, в начале поля стоит трактор, ему необходимо собрать монетки и вспахать все поле. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?

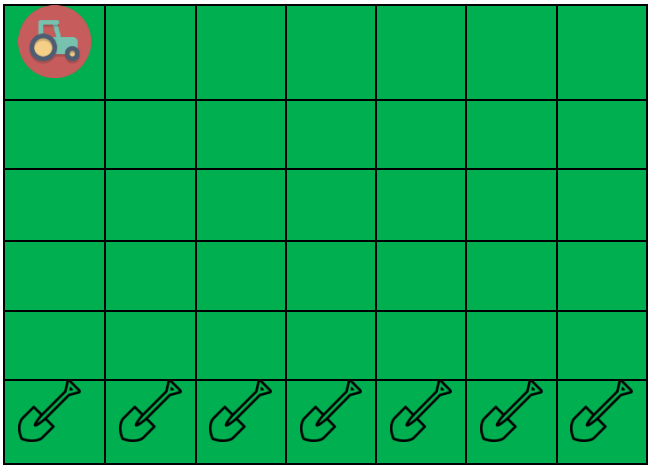


3 уровень – Циклические алгоритмы. Цикл N-раз.

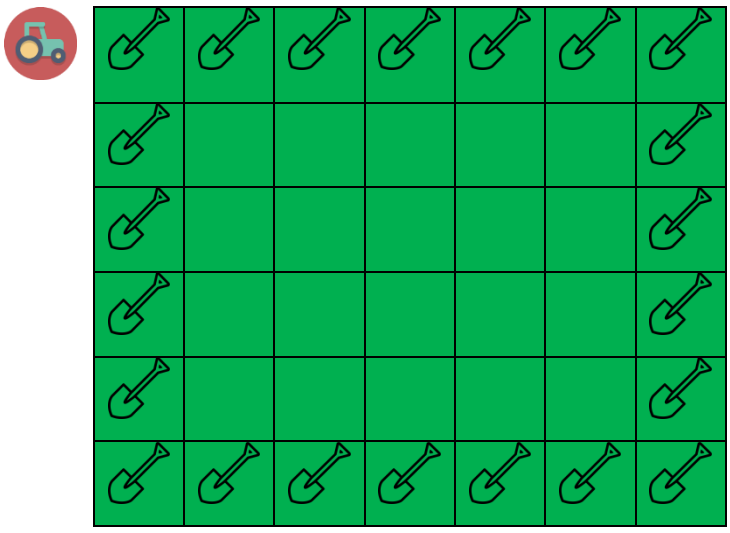
На данном уровне предлагается решить задачи используя конструкцию цикла по параметру. Задачи начинаются с самых простых, где использование данной конструкции будет очевидным и позволит значительно сократить количество действий отводимых на решение задачи, с постепенным усложнением обстановки и количеством необходимых циклов и действий в нем. Образец задач 3 уровня представлен ниже:

Задачи:

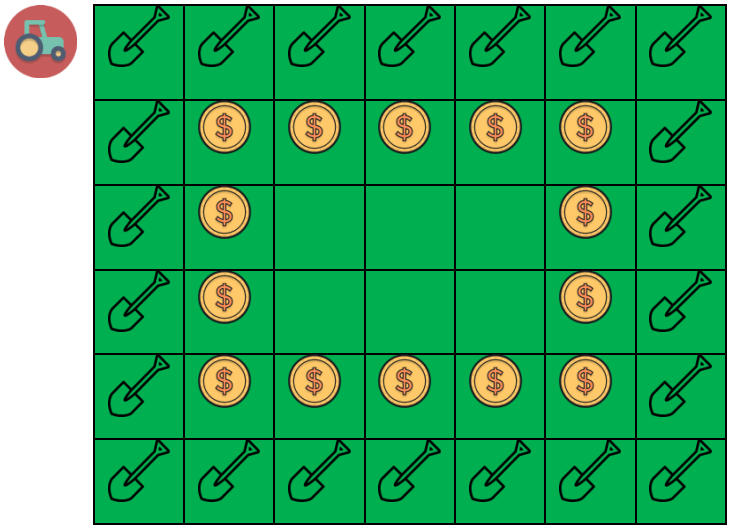
12. Перед вами поле размером 7×6, в начале поля стоит трактор, ему необходимо вспахать поле в обозначенных местах  . Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



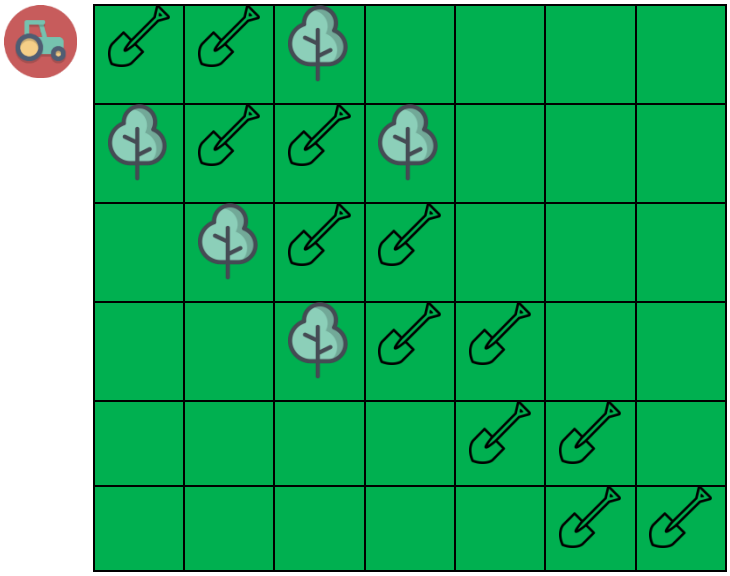
13. Перед вами поле размером 7×6, в начале поля стоит трактор, ему необходимо вспахать поле по контуру. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



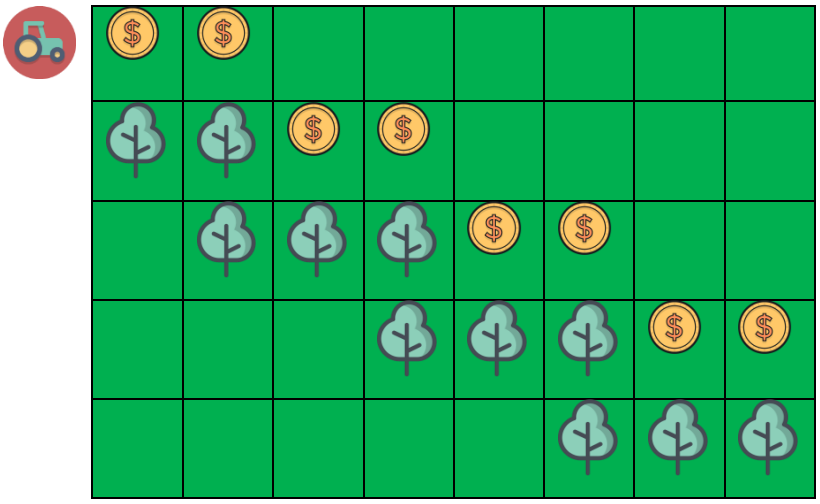
14. Перед вами поле размером 7×6, в начале поля стоит трактор, ему необходимо вспахать поле по контуру и собрать монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



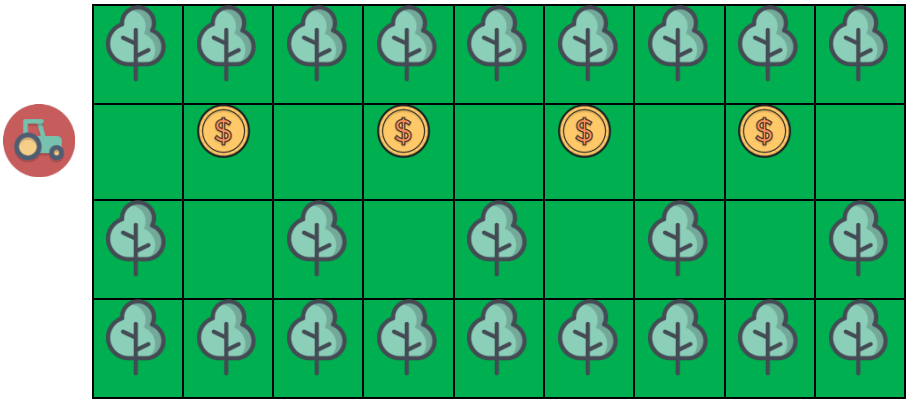
15. Перед вами поле размером 7×6, в начале поля стоит трактор, ему необходимо вспахать поле в обозначенных местах  . Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



16. Перед вами поле размером 8×5, в начале поля стоит трактор, ему необходимо собрать все монетки на поле. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



17. Перед вами поле размером 9×4, в начале поля стоит трактор, ему необходимо собрать монетки и вспахать все поле. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?

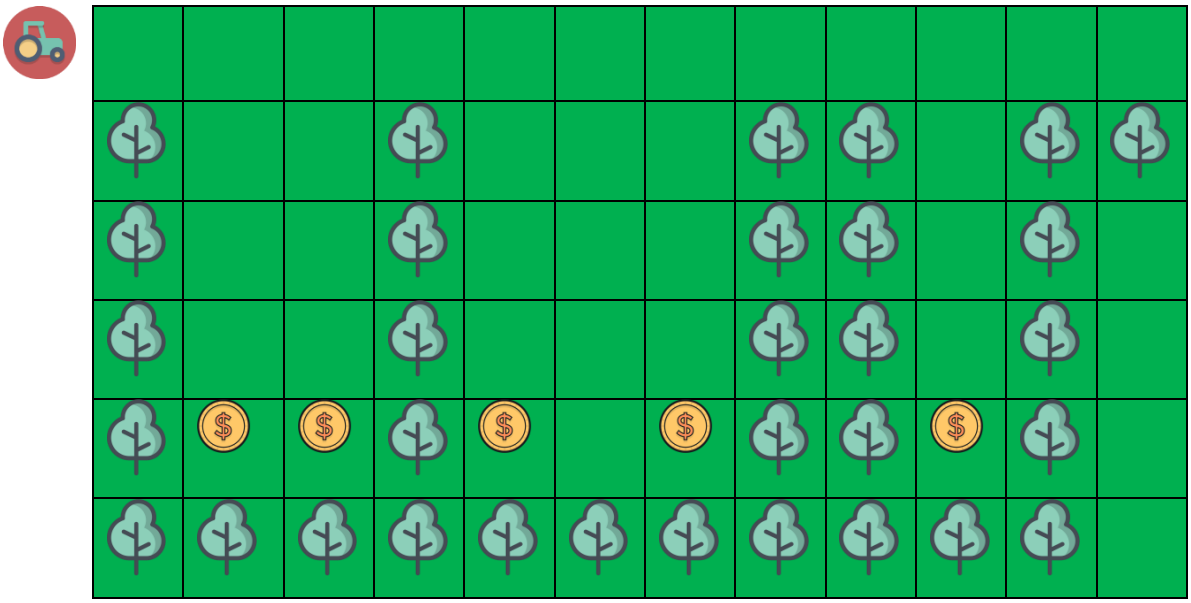


4 уровень – Команды ветвления. Если, выбор.

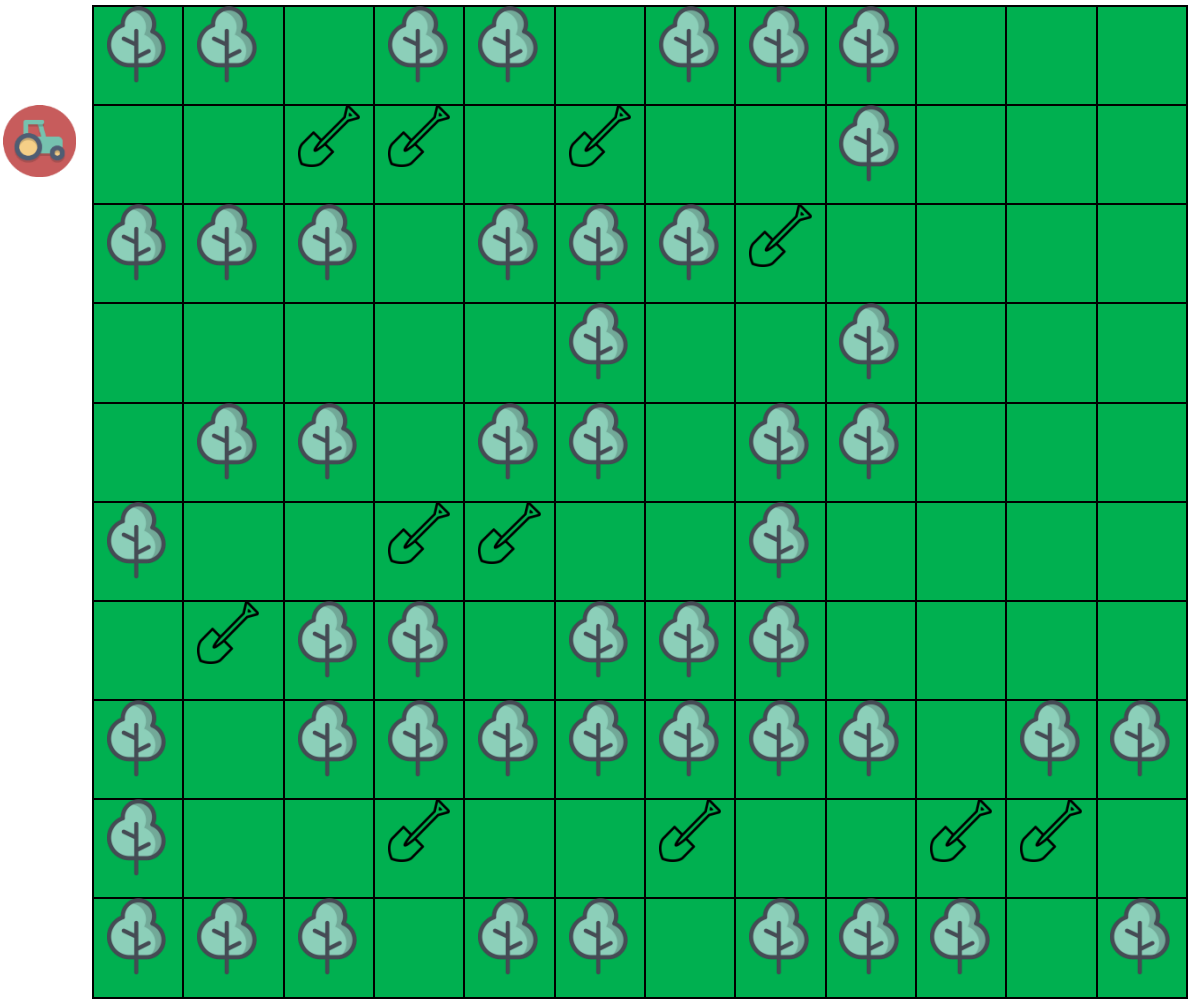
Данный уровень является промежуточным при освоении циклических конструкций. На этом этапе формируются навыки в построении логических выражений и выполнение тех или иных действий в соответствии с условием. Приветствуется объединение однотипных действий в цикл по параметру. Образец задач 4 уровня представлен ниже:

Задачи:

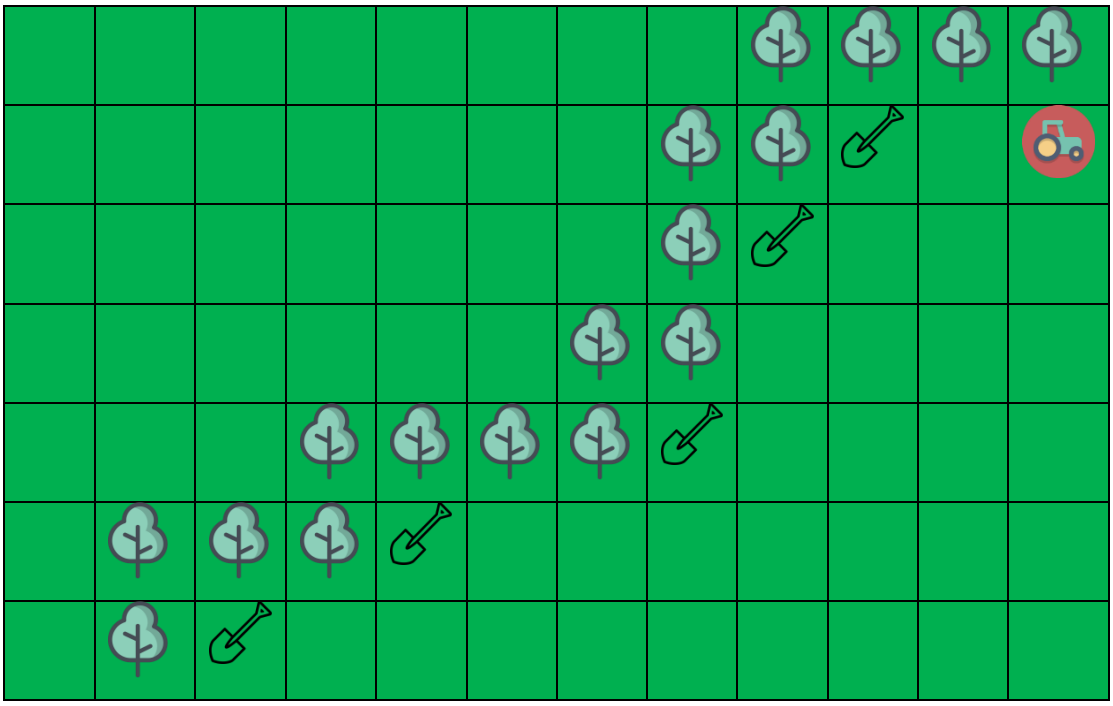
18. Перед вами поле размером 12×6, в начале поля стоит трактор, ему необходимо собрать все монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



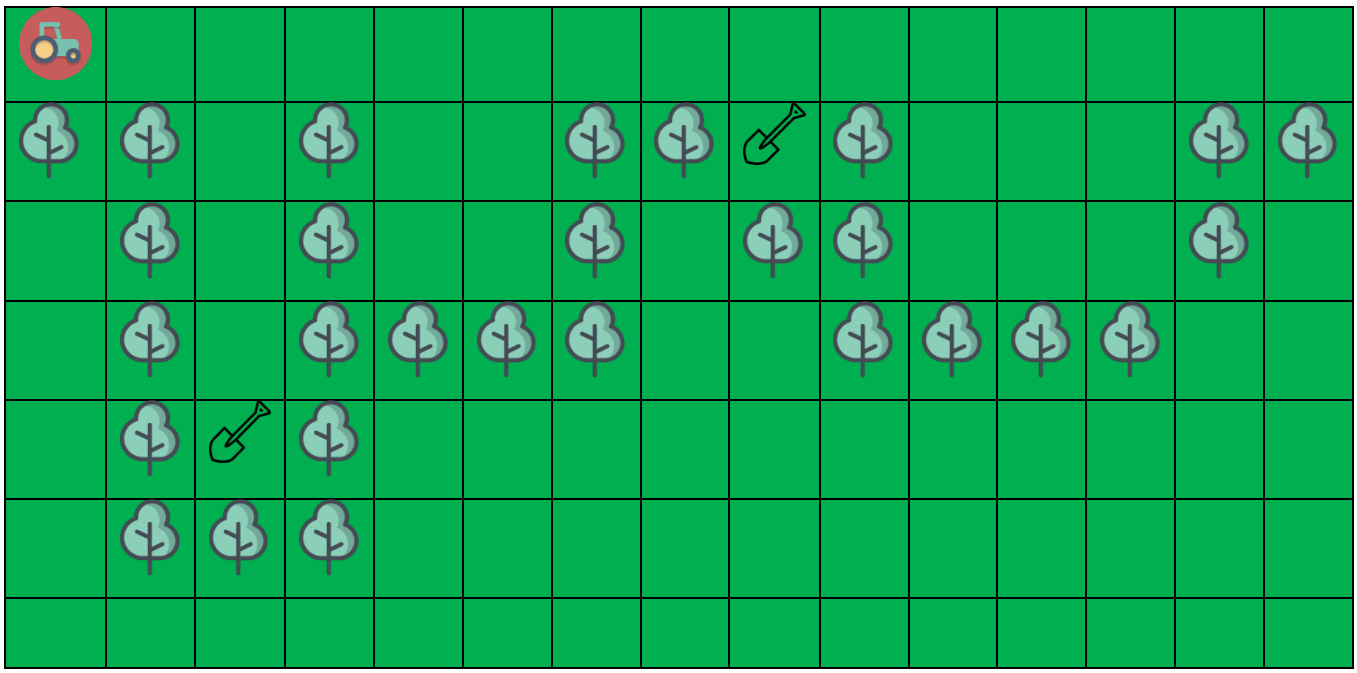
19. Перед вами поле размером 12×10, в начале поля стоит трактор, ему необходимо вскопать поле в обозначенных местах . Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



20. Перед вами поле размером 12×7, в начале поля стоит трактор, ему необходимо вскопать поле в обозначенных местах . Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



21. Перед вами поле размером 12×7, в начале поля стоит трактор, ему необходимо вскопать поле в обозначенных местах  . Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



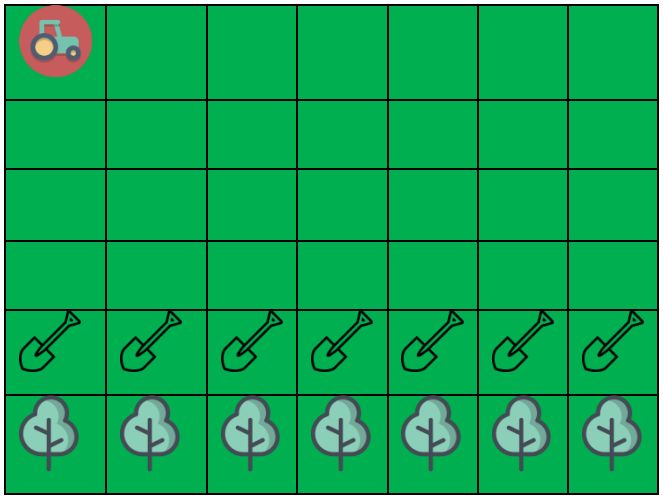
5 уровень – Циклические алгоритмы. Цикл ПОКА.

Следующим этапом в освоении циклических конструкций является Цикл ПОКА. В котором необходимо задавать условия в результате, которого будет выполняться ряд действий. Задачи начинаются с самых простых с постепенным усложнением обстановки и количеством

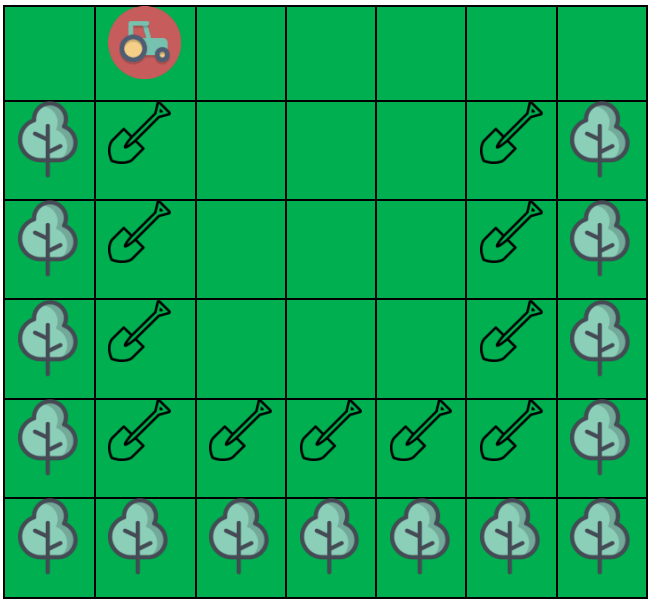
необходимых циклов ПОКА и действий в нем. Образец задач 5 уровня представлен ниже:

Задачи:

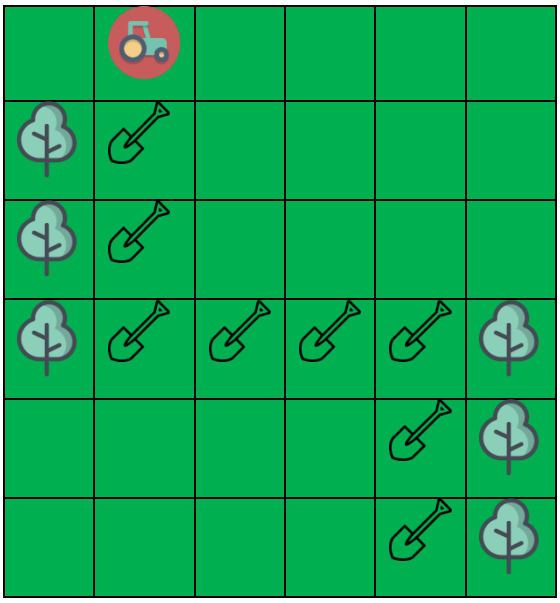
22. Перед вами поле размером 7×6, в начале поля стоит трактор, ему необходимо вспахать поле в обозначенных местах  . Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



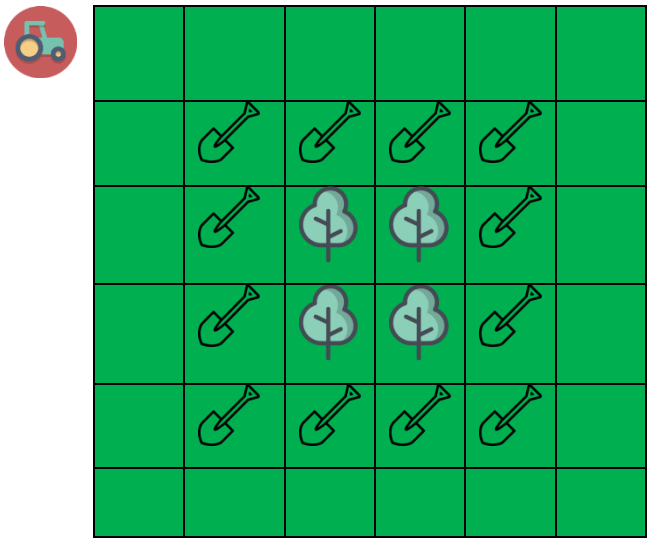
23. Перед вами поле размером 7×6, в начале поля стоит трактор, ему необходимо вспахать поле в обозначенных местах  . Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



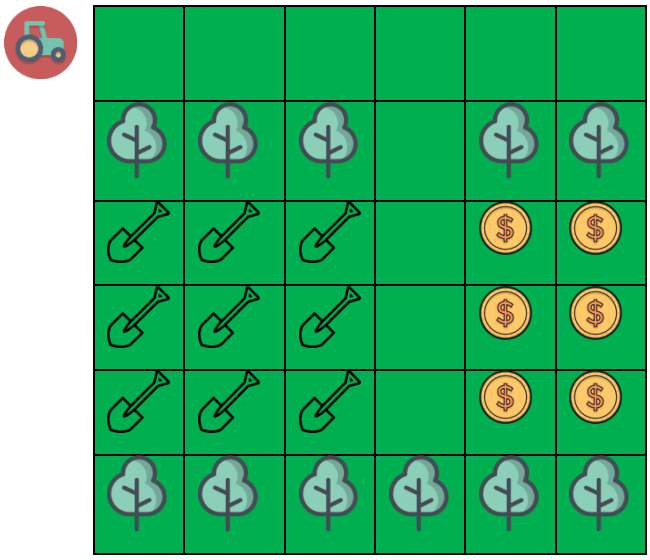
24. Перед вами поле размером 6×6, в начале поля стоит трактор, ему необходимо вспахать поле в обозначенных местах  . Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



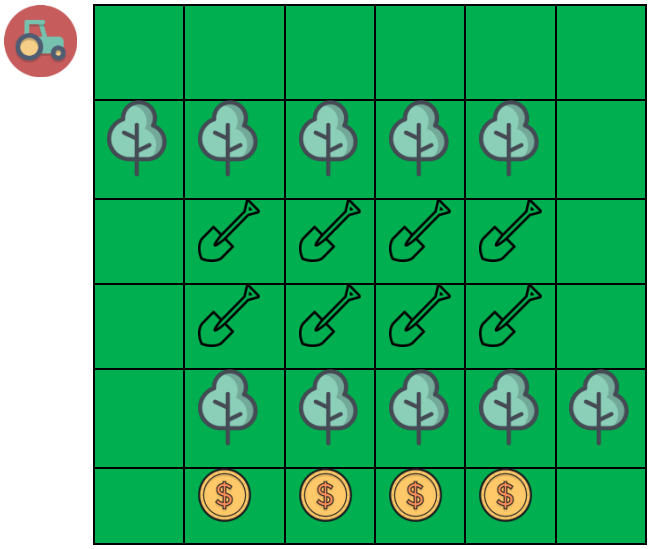
25. Перед вами поле размером 6×6, в начале поля стоит трактор, ему необходимо вспахать поле в обозначенных местах . Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



26. Перед вами поле размером 6×6, в начале поля стоит трактор, ему необходимо вспахать поле в обозначенных местах и собрать монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



27. Перед вами поле размером 6×6, в начале поля стоит трактор, ему необходимо вспахать поле в обозначенных местах  и собрать монетки. Напишите, какие действия должен сделать трактор для выполнения поставленной задачи?



В Приложении А представлен конспект урока по теме: «Основы алгоритмизации и программирования», разработан с учетом методики представленной в пунктах 2.1. и 2.2. данной работы.

2.3. Переход на язык программирования Паскаль

Следующим этапом является непосредственный переход на язык программирования Паскаль, изучение основ и написание собственных программ. Изучение основ программирования происходит в игровой форме с помощью визуального исполнителя и меняющейся обстановкой вокруг него.

Представленная программируемая игра максимально приближена к уже известным для учащихся постановкам, что делает переход к языку программирования точечным, снижает страх перед написанием кода, не вызывает визуального отторжения, как было ранее, при написании программ и выполнении их в консольном окне. Внешний вид игры представлен на рисунке 7.

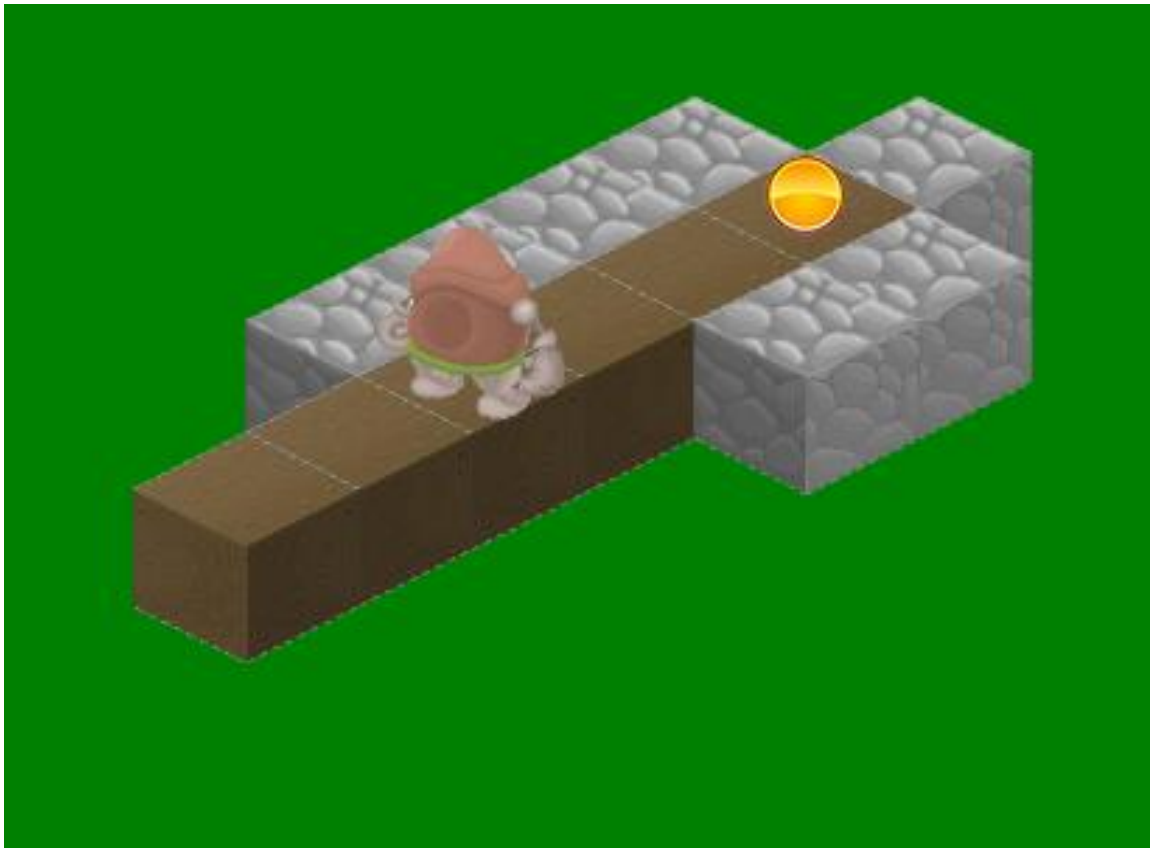


Рисунок 7 - Игра на языке Паскаль

Традиционно программой принято называть упорядоченную последовательность команд или инструкций, которые выполняются компьютером в процессе решения какой-либо задачи [31].

В данной игре учащимся необходимо задавать последовательность действий для персонажа, которые впоследствии образуют код программы, используя различные алгоритмические структуры. Действия доступные персонажу представлены ниже:

- Вперед - позволяет персонажу сделать шаг вперед (***procedure hero_forward)***.
- Налево - позволяет персонажу повернуться налево (***procedure hero_turn_left)***.

– Направо - позволяет персонажу повернуться направо (***procedure hero_turn_right)***.
– Взять - позволяет персонажу взять монетку (***procedure hero_take_coin***).

Выбор действия для персонажа осуществляется путем обращения к уже существующим функциям и во время написания своей программы учащимся необходимо правильно обратиться к фрагментам программы, в результате чего закрепляется навык работы с процедурами и функциями, которые, как правило, формируются плохо т.к. изучение данной темы стоит в конце раздела или не изучается вовсе в связи с ограниченным количеством часов. Пример действий заданных персонажу для решения тестовой задачи представлен на рисунке 8.

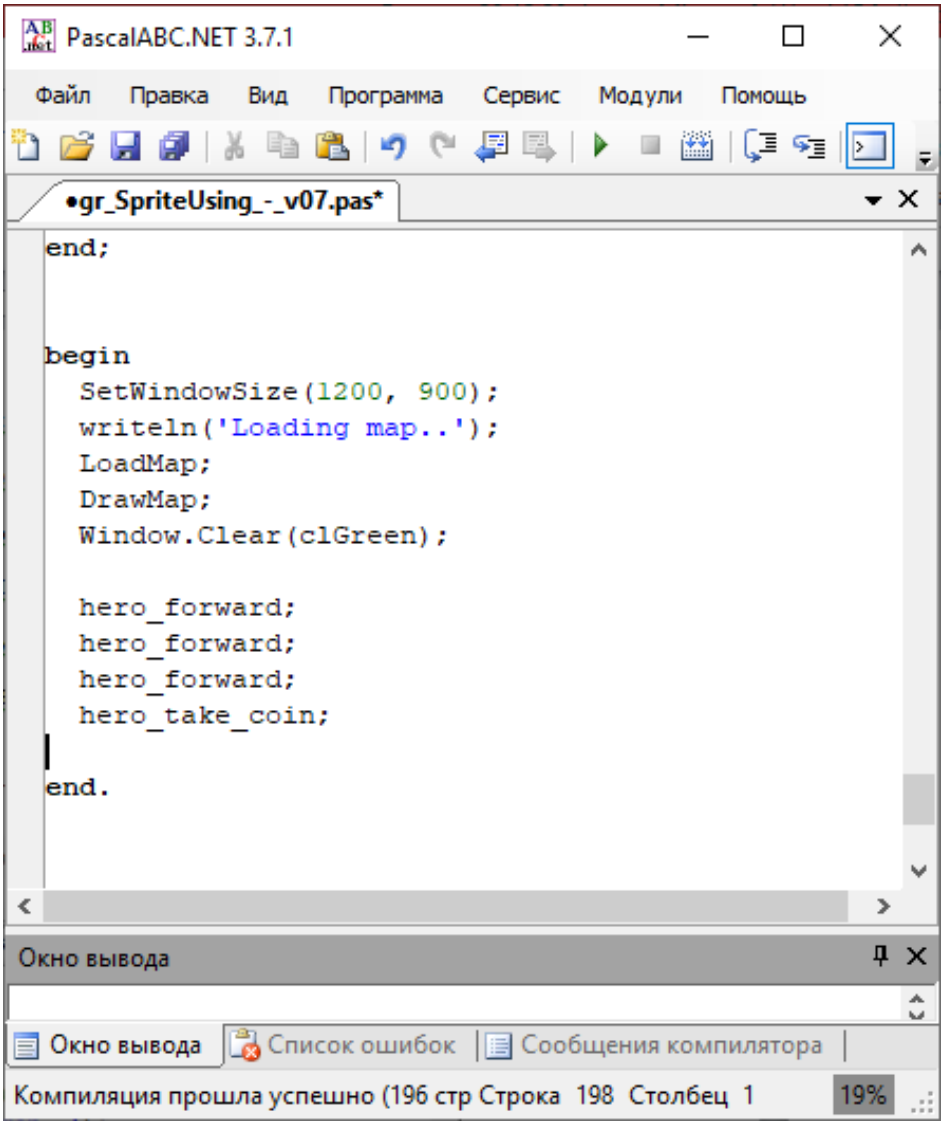


Рисунок 8 – Пример выполнение игры на языке Паскаль

Также в игре действует система проверки ошибок, позволяющая отследить синтаксис и семантику написанной программы учащимся. Проверка происходит стандартными способами среды PascalABC в процессе предварительной компиляции кода.

Игра разработана в среде PascalABC, т.к. интерфейс данной среды интуитивно понятен и не нагружен вспомогательными окнами, что позволяет не отвлекаться и не путаться с дополнительным функционалом программы. Выбранная среда для разработки отлично подойдет для использования в школах и СПО, учителя смогут самостоятельно настраивать обстановку и сложность игры, что делает ее общедоступной и гибкой в использовании.

При прохождения игры учитель может отследить следующие моменты:

- проверить количество шагов выполненных исполнителем для достижения цели и в дальнейшем подобрать задания на решение задачи с минимальным количеством ходов;
- проверить на корректность выполнения условий задачи;
- сохранить траекторию движения исполнителя без анимации, а сразу с прорисовкой траектории при каждом изменении программы;
- сохранить программу учащегося для дальнейшей аналитики и проверки.

Условно реализации задуманной игры была разбита на следующие этапы:

- 1 этап – создание обстановки;
- 2 этап – создание действий персонажа;
- 3 этап – создание анимации.

1 этап – создание обстановки.

Для создания обстановки было решено использовать файловые переменные позволяющие считать информацию находящуюся в текстовом

файле (Блокнот) [32]. В этом текстовом файле хранится карта задачи, где используя условные обозначения, описано расположение персонажа и обстановка которая его окружает. На рисунке 9 представлено описание карты для тестовой задачи.

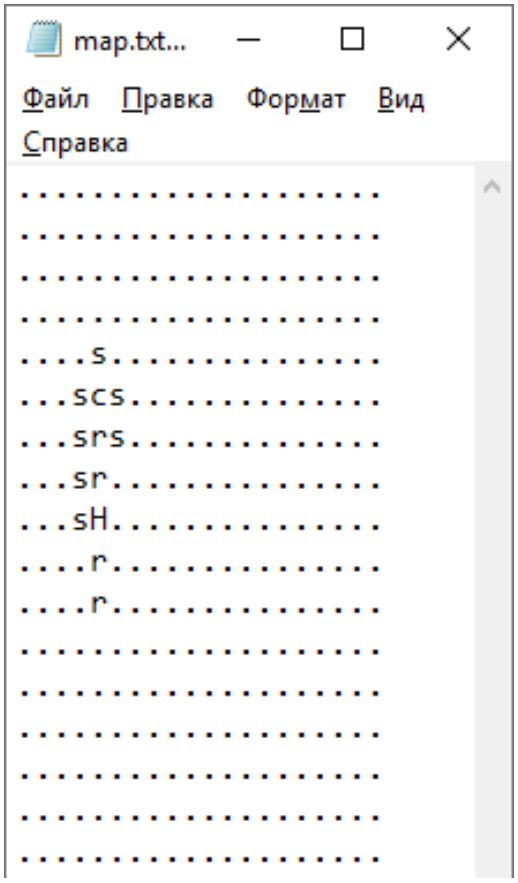


Рисунок 9 - Описание карты в Блокноте

В блокноте используется следующее условное обозначение:

- «s» - каменный блок;
- «r» - земляной блок;
- «c» - монетка;
- «H» - персонаж;
- «.» - пустое пространство.

Исходя из условного обозначения на карте, используя подключенный графический модуль ABCObjects, происходит прорисовка обстановки и затем подгруздка векторных изображений для графического отображения карты.

Код процедуры, позволяющий считать информацию с блокнота представлен ниже.

```
procedure LoadMap;
const map_filename = 'map.txt';
var   i, j      : integer;
      map_file  : file;
      c         : char;
      line      : string;
begin
  Assign(map_file, map_filename);
  Reset(map_file);
  for j := 20 downto 1 do
  begin
    for i := 1 to 20 do
    begin
      read(map_file, c);
      coins_data[i, j] := 0;
      case c of
        '.': map_data[i, j] := VOID;
        's': map_data[i, j] := STONE;
        'r': map_data[i, j] := ROAD;
        'c': begin
               map_data[i, j] := ROAD;
               coins_data[i, j] := 1;
             end;
        'H': begin
               map_data[i, j] := ROAD;
               hero_i := i;
               hero_j := j;
               hero_dir := HERO_WEST;
             end;
      end;
    end;
    read(map_file, c, c);
```

```
  end;
      end;
```

При выполнении процедуры **procedure** LoadMap используются те же условные обозначения объектов карты, что и в блокноте, каждому элементу с помощью цикла присваивается своя координата, в результате образуется сетка из графических объектов.

После того как карта с блокнота загружена в программу необходимо подгрузить векторных изображений для графического отображения карты, для этого создана следующая процедура:

```
procedure DrawMap;
var   i, j      : integer;
begin
  for i := 1 to 20 do
  for j := 20 downto 1 do
  begin
    case map_data[i, j] of
      STONE: map_pict[i, j] := new PictureABC(scr_x(i, j), scr_y(i, j),
'tile_stone.png');
      ROAD: map_pict[i, j] := new PictureABC(scr_x(i, j), scr_y(i, j),
'tile_road.png');
    end;
    if coins_data[i, j] = 1 then
      coins_pict[i, j] := new PictureABC(scr_x(i, j) + TILE_DIAG div 2 -
COIN_SIZE div 2,  scr_y(i, j),  'heart-coins.spinf');
  end;
  sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_N.png');
      end;
```

Для графического отображения карты используются следующие векторные изображения:

- каменный блок представлен на рисунке 10;

Рисунок 10 - Каменный блок

– земляной блок представлен на рисунке 11;

Рисунок 11 - Земляной блок

– монетка представлена на рисунке 12;

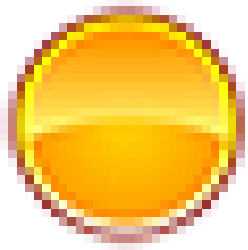


Рисунок 12 - Монетка

– персонаж представлен на рисунке 13;

Рисунок 13 - Раскадровка персонажа

Стоит отметить особенности работы с графическим окном в программе PascalABC, расположение осей отличаются от привычных всеми со школьного курса математики, начало координат находится в левом верхнем углу и ось *X* увеличивается с право налево, а ось *Y* увеличивается сверху вниз, пример оси координат представлен на рисунке 14.

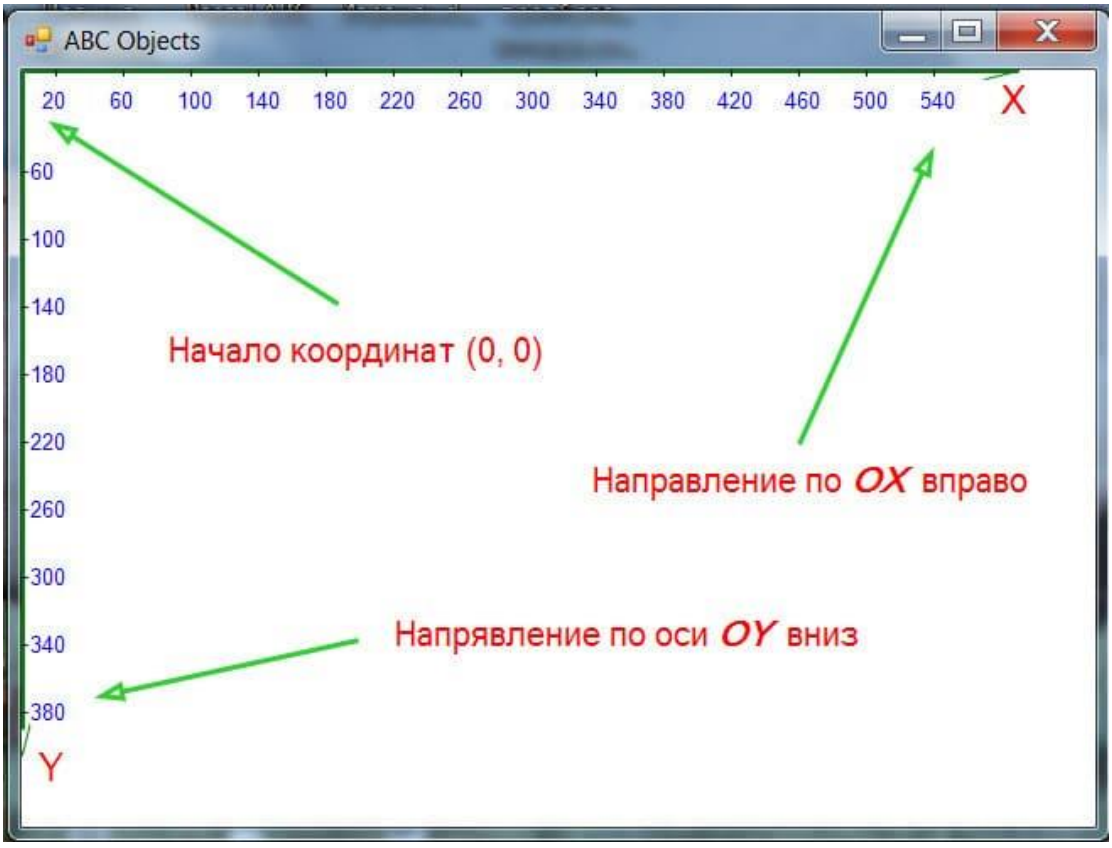


Рисунок 14 – Ось координат в графическом окне

2 этап – создание действий персонажа.

Для того чтобы персонаж мог выполнять заданные ему действия были прописаны следующие функции:

– ***procedure hero_forward*** - позволяет персонажу сделать шаг вперед. Текст функции представлен ниже.

```
procedure hero_forward;
begin
  case hero_dir of
    HERO_NORTH: inc(hero_j);
    HERO_SOUTH: dec(hero_j);
    HERO_EAST:  inc(hero_i);
    HERO_WEST:  dec(hero_i);
  end;
  sprite_hero.MoveTo(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2);
end;
```

При работе процедуры ***hero_forward*** происходит перерасчет координат в зависимости от расположения и направления персонажа, если персонаж направлен на Север (состояние HERO_NORTH), то увеличивается координата по оси «j» (ось Y на рисунке 13) и персонаж совершает шаг вперед в сторону южного направления по отношению оси координат.

Если персонаж направлен на Юг (состояние HERO_SOUTH), то уменьшается координата по оси «j» (ось Y на рисунке 13) и персонаж совершает шаг назад в сторону южного направления по отношению оси координат.

Если персонаж направлен на Восток (состояние HERO_ EAST), то увеличивается координата по оси «i» (ось X на рисунке 13) и персонаж совершает шаг вправо в сторону восточного направления по отношению оси координат.

Если персонаж направлен на Запад (состояние HERO_ WEST), то уменьшается координата по оси «i» (ось X на рисунке 13) и персонаж совершает шаг вправо в сторону западного направления по отношению оси координат.

– ***procedure hero_turn_right*** - позволяет персонажу повернуться направо. Текст функции представлен ниже.

```
procedure hero_turn_right;
begin
  sprite_hero.Destroy;
  case hero_dir of
HERO_NORTH:
    begin
      hero_dir := HERO_EAST;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_E.png');
    end;
 HERO_SOUTH:
    begin
      hero_dir := HERO_WEST;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_W.png');
    end;
HERO_EAST:
    begin
      hero_dir := HERO_SOUTH;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_S.png');
    end;
HERO_WEST:
    begin
```

```
    hero_dir := HERO_NORTH;
    sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_N.png');
  end;
 end;
end;
```

Для поворота персонажа считывается его текущее направление по оси координат (сторон света) и относительно него происходит поворот направо по средствам перерисовки состояние персонажа.

Если персонаж направлен Север (состояние HERO_NORTH), то ему присваивается направление на Восток (состояние HERO_EAST) и загружается новое изображение персонажа.

Если персонаж направлен Юг (состояние HERO_ SOUTH), то ему присваивается направление на Запад (состояние HERO_ WEST) и загружается новое изображение персонажа.

Если персонаж направлен Восток (состояние HERO_ EAST), то ему присваивается направление на Юг (состояние HERO_ SOUTH) и загружается новое изображение персонажа.

Если персонаж направлен Запад (состояние HERO_ WEST), то ему присваивается направление на Север (состояние HERO_ NORTH) и загружается новое изображение персонажа.

– ***procedure hero_turn_left*** - позволяет персонажу повернуться налево. Текст функции представлен ниже.

```
procedure hero_turn_left;
begin
 sprite_hero.Destroy;
 case hero_dir of

HERO_NORTH:
  begin
```

```
      hero_dir := HERO_WEST;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_W.png');
    end;
HERO_EAST:
    begin
      hero_dir := HERO_NORTH;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_N.png');
    end;
HERO_SOUTH:
    begin
      hero_dir := HERO_EAST;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_E.png');
    end;
 HERO_WEST:
    begin
      hero_dir := HERO_SOUTH;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_S.png');
    end;
  end;
end;
```

Поворот персонажа налево происходит аналогичным способом, как и поворот направо, считывается его текущее направление по оси координат (сторон света) и относительно него происходит поворот по средствам перерисовки состояние персонажа.

Если персонаж направлен Север (состояние HERO_NORTH), то ему присваивается направление на Запад (состояние HERO_ WEST) и загружается новое изображение персонажа.

Если персонаж направлен Восток (состояние HERO_ EAST), то ему присваивается направление на Север (состояние HERO_ NORTH) и загружается новое изображение персонажа.

Если персонаж направлен Юг (состояние HERO_ SOUTH), то ему присваивается направление на Восток (состояние HERO_ EAST) и загружается новое изображение персонажа.

Если персонаж направлен Запад (состояние HERO_ WEST), то ему присваивается направление на Юг (состояние HERO_ SOUTH) и загружается новое изображение персонажа.

– ***procedure hero_take_coin*** - позволяет персонажу взять монетку. Текст функции представлен ниже.

```
procedure hero_take_coin;
begin
  if coins_data[hero_i, hero_j] <> 0 then
  begin
    coins_data[hero_i, hero_j] := 0;
    coins_pict[hero_i, hero_j].Destroy;
  end
  else
    writeln('ERROR!');
end;
```

Данная функция работает в следующем порядке, если персонаж находится на том же блоке что и монетка, и персонаж может ее «взять», в этом случаи монетка с карты исчезает и персонаж продолжает выполнять действия заданные учащимся. Если команда персонажу была задана не верно, т.е. монетки перед ним нет, то учащийся увидит сообщение об ошибке (рис. 15).

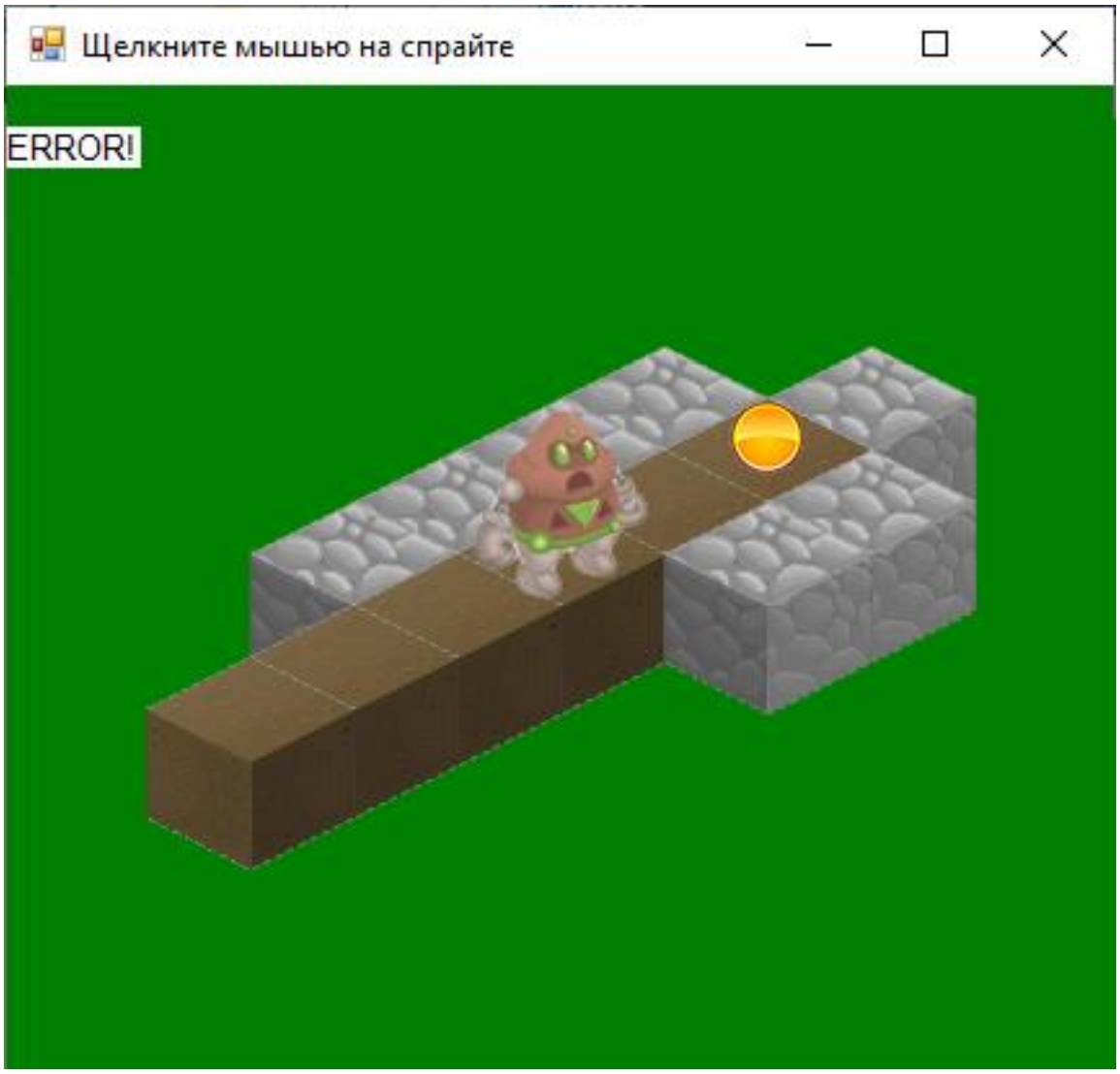


Рисунок 15 – Сообщение об ошибке

3 этап – создание анимации.

Одна из самых сложных задач при создании игры стало создание эффекта анимации, достичь которого можно используя модули рисования с учётом перекрытий картинок и анимации с перерисовкой в выбранной среде PascalABC.

Реализация игры выполнена с использованием модуля ABCSprites. Данный модуль позволяет создать анимационные объекты с автоматически меняющимися кадрами [33].

Модуль работает с растровыми изображениями и чтобы создать эффект анимации нужно сделать набор раскадровок, который называются атласом спрайтов [34]. На рисунке 16 представлен пример раскадровки монеты.

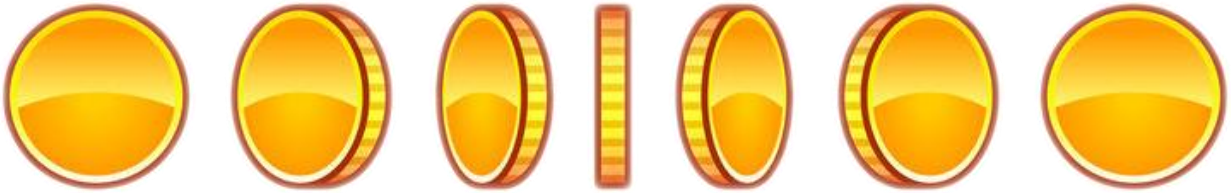


Рисунок 16 – Раскадровка монеты

Для создания эффекта анимации работа атласа спрайта описывается в текстовом документе (Блокноте), в котором задается ширина и количество кадров, скорость анимации спрайт, количество состояний объекта [33]. На рисунке 17 представлено описание спрайта для анимации монеты.

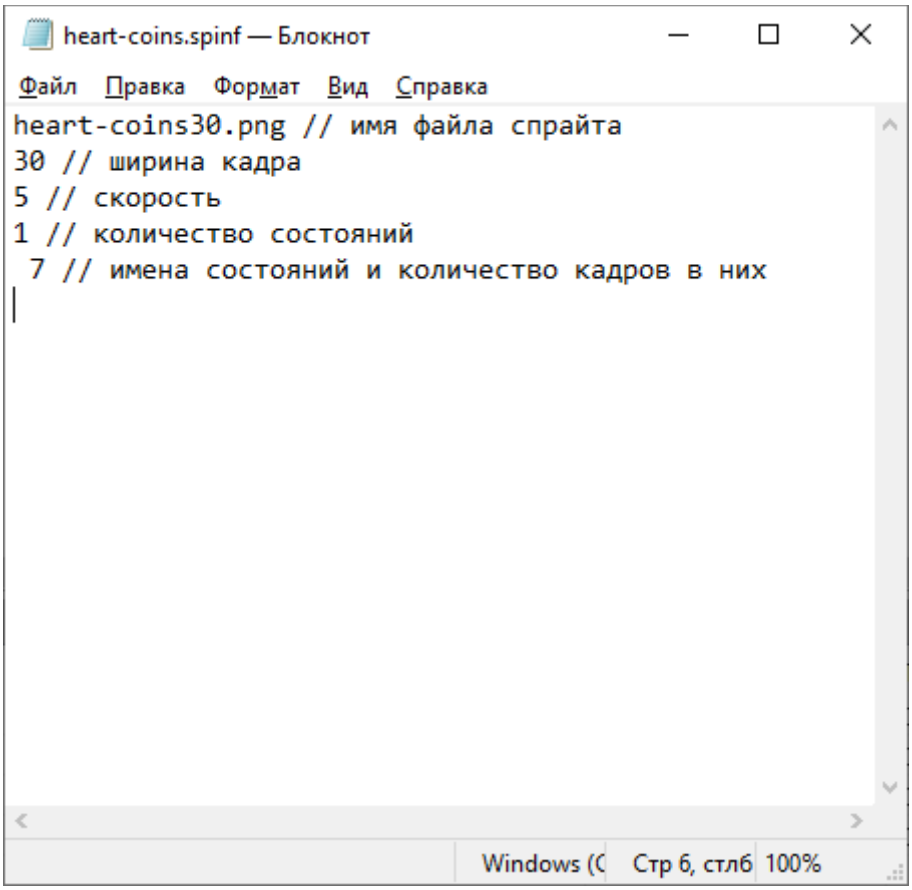


Рисунок 17 – Анимация монеты

Полный код игры представлен в Приложении Б.

ЗАКЛЮЧЕНИЕ

В результате выполнения выпускной квалификационной работы был проведён анализ существующих методик преподавания программирования в школе и СПО. Проанализированы проекты, основанные на мотивации обучающихся игровым, групповым, творческим, соревновательным и социальным методикам и методами продвижения.

В результате анализа было выявлено, что игровые технологии способствуют поддержки интереса учащихся к изучаемому материалу, достаточному для активизации и вовлечению ученика в учебный процесс самостоятельно.

Также в ходе работы был выполнен обзор литературы по теме исследования и смежным темам.

Была разработана методика обучения языку программирования Паскаль студентов СПО, нацеленная на формирование долгосрочных остаточных знаний, необходимых и достаточные для практического применения в профессиональной деятельности и дальнейшего развития при обучении в учреждениях более высокого уровня.

Представленная методика состоит из трех этапов:

1 этап – квест-игра «Algorithm City». Учащиеся получают базовые концепции кодирования, такие как последовательность команд, функции и циклы (петли), путем управления персонажем и выполнением действий в виде сбора золотых монет и решение уровней.

2 этап – Самостоятельная работа. Данный этап позволяет осуществить контроль сформированных навыков полученных при прохождении игры и закрепление их на практике.

Для данного этапа был разработан сборник задач для самостоятельной работы, решение которых необходимо выполнить в письменном виде, что позволяет отсечь интуитивное прохождение игры, и делает обучение более осознанным.

3 этап – переход на язык программирования Паскаль. Изучение основ программирования происходит в игровой форме с помощью визуального исполнителя и меняющейся обстановкой вокруг него.

Для реализации третьего этапа спроектировали систему уроков и разработали визуальную среду исполнителя позволяющую отследить, процесс исполнения, и результат выполнения алгоритма. Разработана система автоматической проверки задания и визуализацией ошибки, а также система проверки соблюдения дополнительных ограничений и визуализации их нарушения.

Если сохранить контекст, в котором формировались понятия условного оператора, исполнителя, алгоритмов и конструкций, то можно перейти к языку программирования Паскаль, подав его как язык программирования исполнителя, что позволяет:

- геймифицировать процесс знакомства с программированием с использованием BYOD;
- скомпенсировать отсутствие аудиторных часов;
- формировать долгосрочные (остаточные) знания;
- выполнить плавную смену контекста на программирование в профессиональной IDE, в которой завершается формирование требуемых компетенций.

СПИСОК ИСПОЛЬЗУЕМЫХ ИСТОЧНИКОВ

1. Прогрмма подготовки специалистов среднего звена по специальности 35.02.07 "Механизация сельского хозяйства" URL: [http://vtalk-vbg.ru/org-info/education-program?id=23] (дата обращения: 20.12.2019);

2. Новицкий И. «Точное земледелие: принцип работы и перспективы» URL: [https://xn--80ajgpcpbhkds4a4g.xn--p1ai/articles/tochnoe-zemledelie/] (дата обращения: 20.12.2019);

3. Агротехнологии «Global Position System, или по-нашему - Джи Пи Эс Система глобального позиционирования (GPS) URL: [http://agrotehnology.com/tochnoe-zemledelie/ideologi/global-position-system-ili-po-nashemu-dzhi-pi-es-sistema-globalnogo-pozicionirovaniya-gps] (дата обращения: 20.12.2019);

4. Esri-cis Блок «Геоинформационные системы в сельском хозяйстве». URL: [https://blogs.esri-cis.ru/2018/08/09/gis-for-agriculture/] (дата обращения: 20.12.2019);

5. Википедия «Дистанционное зондирование Земли». URL: [https://ru.wikipedia.org/wiki/Дистанционное_зондирование_Земли] (дата обращения: 20.12.2019);

6. «БПЛА как основа земледелия ближайшего будущего» URL: [https://iot.ru/selskoe-khozyaystvo/bpla-kak-osnova-zemledeliya-blizhayshego-budushchego] (дата обращения: 20.12.2019);

7. Яндекс.Лицей URL: [https://yandexlyceum.ru/] (дата обращения: 20.12.2019);

8. IT ШКОЛА SAMSUNG URL: [https://www.samsung.com/ru/itschool/about/] (дата обращения: 20.12.2019);

9. Олимпиада НТИ. URL: [https://nti-contest.ru/profiles/] (дата обращения: 20.12.2019);

10. Конюхова К. «Час кода 2020: история и традиции праздника» URL: [https://www.kp.ru/putevoditel/obrazovanie/chas-koda/] (дата обращения: 20.12.2019);

11. Jordan Peterson | The Most Terrifying IQ Statistic URL: [https://www.youtube.com/watch?v=5-Ur71ZnNVk] (дата обращения: 20.12.2019);

12. Кушниренко, А. Г., and Г. В. Лебедев. «Информатика: 12 лекций о том, для чего нужен школьный курс информатики и как его преподавать» Лаборатория Базовых Знаний (2000);

13. «Система программирования КуМир» URL: [https://www.niisi.ru/kumir/] (дата обращения: 20.12.2019);

14. Wolfram, S. (2002). A new kind of science (Vol. 5, p. 130). Champaign, IL: Wolfram media.

15. Шарипов Фаридун Файзуллаевич, Мараджабов Собирджон Исоматович Теоретическая модель формирования алгоритмического мышления студентов вузов в процессе обучения объектно-ориентированному программированию // БГЖ. 2017. №3 (20). URL: [https://cyberleninka.ru/article/n/teoreticheskaya-model-formirovaniya-algoritmicheskogo-myshleniya-studentov-vuzov-v-protsesse-obucheniya-obektno-orientirovannomu] (дата обращения: 20.12.2020);

16. Берман Нина Демидовна Роль информационных технологий в развитии навыков вычислительного мышления // Мир науки. Педагогика и психология. 2019. №2. URL: [https://cyberleninka.ru/article/n/rol-informatsionnyh-tehnologiy-v-razvitii-navykov-vychislitelnogo-myshleniya (дата обращения: 01.12.2020)];

17. Еремеева Наталья Николаевна Формирование алгоритмического мышления у школьников в ходе групповой работы // Пермский педагогический журнал. 2013. №4. URL: [https://cyberleninka.ru/article/n/formirovanie-algoritmicheskogo-myshleniya-u-shkolnikov-v-hode-gruppovoy-raboty (дата обращения: 01.12.2020)];

18. И. Р. Дединский, Кафедра информатики МФТИ, Москва «Аналитический подход к довузовскому преподаванию программирования» URL: [https://habr.com/ru/post/179307/];

19. Пушкарева Т. П., Калитина В. В. Визуализированная методика обучения программированию //Современные проблемы науки и

образования. – 2014. – №. 5. – С. 31-31. URL: [https://www.science-education.ru/pdf/2014/5/193.pdf (дата обращения: 01.12.2020)];

20. Родыгин Евгений Федорович Методические рекомендации обучения программированию в школе // Вестник Марийского государственного университета. 2011. №7. URL: [https://cyberleninka.ru/article/n/metodicheskie-rekomendatsii-obucheniya-programmirovaniyu-v-shkole (дата обращения: 01.12.2020)];

21. Доронина К.Е, Шимов И.В. «Методика обучения программированию с использованием исполнителей с обратной связью на уроках информатики в школе» 2016 [http://elar.uspu.ru/bitstream/uspu/3064/1/03Doronina2.pdf (дата обращения: 23.11.2020)];

22. А.П. Ершов и школьная информатика. Курс «История информатики» Лекция URL: [https://zodorov.ru/kurs-istoriya-informatiki-lekciya.html?page=17(дата обращения: 23.11.2020)];

23. Зубков Олег Владимирович, Семичева Наталья Леонидовна Роль автоматизированных тестирующих систем в процессе формирования навыков алгоритмического мышления // Педагогический ИМИДЖ. 2019. №4 (45). URL: [https://cyberleninka.ru/article/n/rol-avtomatizirovannyh-testiruyuschih-sistem-v-protsesse-formirovaniya-navykov-algoritmicheskogo-myshleniya (дата обращения: 01.12.2020)];

24. Якименко Ольга Викторовна, Стась Андрей Николаевич Применение обучающих программ-тренажеров в обучении программированию // Вестник ТГПУ. 2009. №1. URL: [https://cyberleninka.ru/article/n/primenenie-obuchayuschih-programm-trenazherov-v-obuchenii-programmirovaniyu (дата обращения: 01.12.2020)];

25. Дацун Н. Н. Как организовать самостоятельную работу при обучении программированию. – 2000. URL: [http://ea.donntu.edu.ua/bitstream/123456789/5166/3/new%20collegium-2.pdf (дата обращения: 23.11.2020)];

26. Стась А. Н., Прусских О. Н. Формирование алгоритмического мышления в процессе обучения теории графов //Вестник Томского

государственного педагогического университета. – 2012. – №. 2. – С. 166-169. URL: [http://vestnik.tspu.edu.ru/files/vestnik/PDF/articles/stas-_a._n._166_169_2_117_2012.pdf (дата обращения: 23.11.2020)];

27. Родыгин Евгений Федорович Методические рекомендации обучения программированию в школе // Вестник Марийского государственного университета. 2011. №7. URL: [https://cyberleninka.ru/article/n/metodicheskie-rekomendatsii-obucheniya-programmirovaniyu-v-shkole (дата обращения: 01.12.2020)];

28. Паскова Анна Александровна Мобильное обучение в высшем образовании: технологии BYOD // Вестник Майкопского государственного технологического университета. 2018. №4. URL: [https://cyberleninka.ru/article/n/mobilnoe-obuchenie-v-vysshem-obrazovanii-tehnologii-byod (дата обращения: 23.11.2020)].

29. Algorithm City : Coding Game URL: [https://play.google.com/store/apps/details?id=air.MusterenGames.ElHarezmiCoding&hl=en_US&gl=US (дата обращения: 20.12.2020)]

30. Андрющенко А.В. «Методика преподавания программирования в основной школе» URL: [https://nsportal.ru/shkola/informatika-i-ikt/library/2015/12/17/metodika-prepodavaniya-programmirovaniya-v-osnovnoy];

31. «Классификация программного обеспечения» URL: [https://gigabaza.ru/doc/26658.html (дата обращения: 20.12.2020)];

32. «Работа с файлами в паскале» URL: [https://itteach.ru/pascal/rabota-s-faylami-v-paskale (дата обращения: 20.12.2020)];

33. Коллектив РуБоард ВикиЧтение «Описание языка PascalABC.NET.
Модуль ABCSprites» URL: [https://it.wikireading.ru/44726 (дата обращения: 20.12.2020)];

34. Свободной энциклопедии «Википедии» Спрайт (компьютерная графика) URL:

[https://ru.wikipedia.org/wiki/Спрайт_(компьютерная_графика) (дата обращения: 20.12.2020)];

35. Кислова Д.А. «Фундаментальная роль математики при изучении алгоритмизации и программирования» Информационные технологии в математике и математическом образовании: материалы VIII Всероссийской научно-методической конференции с международным участием. Красноярск, 13–14 ноября 2019 г. [Электронный ресурс] / отв. ред. В.Р. Майер; ред. кол. – Электрон. дан. / Краснояр. гос. пед. ун-т им. В.П. Астафьева. – Красноярск, 2019.

36. Кислова Д.В. «Применение достижений компьютерных наук в сельском хозяйстве» Актуальные проблемы информатики и информационных технологий в образовании: материалы Всероссийской конференции с международным участием. Красноярск, 23 апреля 2019 г. [Электронный ресурс] / отв. ред. П.С. Ломаско; ред. кол.; – Электрон. дан. / Краснояр. гос. пед. ун-т им. В.П. Астафьева. – Красноярск, 2019.

ПРИЛОЖЕНИЯ

Приложение А

Конспект урока

«Основы алгоритмизации и программирования»

Тема урока: «Алгоритмы и способы их описания. Типы данных»

Цели урока:

- ***образовательные:*** познакомить учащихся с понятием алгоритма, исполнителями алгоритмов, примерами алгоритмов в жизни, алгоритмическим способом решения задач, типы данных.
- ***развивающие:*** развивать у учащихся логическое мышление, память, смекалку, навыки самоконтроля, интерес к предмету; Научить фиксировать когнитивно сложную задачу на каком-либо носителе в виде, позволяющем адресовать части задачи по отдельности без потери полной картины задачи, и для понимания самой задачи в целом.
- ***воспитательные:*** воспитывать творческий интерес к изучению нового материала, аккуратность и точность, популяризация программирования у учащихся.

Тип урока: комбинированный.

Материалы и оборудование: мультимедийный проектор, компьютеры с установленной средой программирования Pascal ABC, опорные конспекты.

План урока:

1. Организационный момент - 1 мин.
2. Мотивация изучения нового материала – 10 мин.
3. Изучение нового материала - 45мин.
4. Самостоятельная работа - 35 мин.
5. Подведение итогов занятия - 2 мин.
6. Домашнее задание - 2 мин.

ХОД УРОКА

1. Организационный момент - 1 минута

Проверка готовности к уроку. Перекличка.

2. Мотивация изучения нового материала – 10 минут

Пример алгоритма на экране:

Машина открыта? **Если да, то**

Сидишь в машине? **Если да, то**

Зажигание работает? **Если да, то**

Зеркала бокового и заднего вида настроены? **Если да, то**

Пристегнут? **Если да, то**

Фары горят? **Если да, то**

Машина прогрелась? **Если да, то**

Можешь ехать!

Нет: Прогрей машину

Нет: Включи фары. **Если да, то**

- Фара ближнего света. Поверните крайнее кольцо рычага. На приборном щитке загорелся лампочка**? Если да, то** перейди к следующему шагу.

  Нет: *повтори действие*

- Фара дальнего света. Потяните рычаг подрулевого переключателя на себя. На приборном щитке загорелся лампочка**? Если да, то** перейди к следующему шагу.

  Нет: *повтори действие*

Нет: *проверь заряд аккумулятора*

Нет: Пристегнись

Нет: Настрой зеркала. **Если да, то**

- Боковое левое зеркала 1. сесть на водительское сиденье; 2. повернуть голову налево; 3. поворачивайте зеркало. Увидели в нем небольшую часть заднего крыла авто? **Если да, то** перейдите к следующему зеркалу.

  Нет: *повторите действие.*

- Боковое правое зеркало 1. голову повернуть к центру машины; 2. повторить действия аналогичные с левым зеркалом.
- Заднее зеркало поверните зеркало. Центр зеркала совпал с центральной точкой заднего стекла? Если да, то перейдите к следующему шагу.

  Нет: *повторите действие.*

Нет: повторите действия по настройке зеркал.

Нет: заведи машину. **Если да, то**

- Вставить ключ в зажигания.
- Повернуть ключ в зажигании.
- Дождаться когда на приборной панели загорятся и погаснут значки: “АВС”, “ЧЕК”, “масло”.
- Повернуть ключ зажигания далее.
- Услышав звук работы двигателя отпустить ключ.

Нет: *проверь заряд аккумулятора*

Нет: сядь в машину

Нет: открыть машину

Что вы можете сказать глядя на этот текст? Как думаете, на, что это похоже и какую задачу выполняет? *(Ответ: Похоже на план необходимых действий для того чтобы поехать на машине)*

Верно, перед вами инструкция или список действий, которые нужно совершить, чтобы завести машину. Подробность описания которой зависит от того кому она предназначена, если новичку, то составлять нужно очень подробно, если опытному водителю, то инструкция может сократиться до пары строк.

3. Изучение нового материала – 45 минут

Каждая инструкция должна обладать следующими **свойствами**:

1. ***Дискретность*** **–** это свойство предполагает, что любой алгоритм должен состоять из последовательности шагов, следующих друг за другом. Следующий шаг выполняется только по завершении предыдущего.
2. ***Понятность (Детерминированность)*** **–** это свойство указывает, что любое действие в алгоритме должно быть строго и недвусмысленно определено и описано для каждого случая.
3. ***Массовость*** **–** это свойство подразумевает, что один и тот же алгоритм может применяться для решения целого класса задач, отличающихся исходными данными.
4. ***Результативность*** требует, чтобы в алгоритме не было ошибок, т.е. при точном исполнении всех команд результат получен для каждой ситуации. Процесс решения задачи должен прекратиться за конечное число шагов и при этом должен быть получен определенный постановкой задачи результат (ответ).
5. ***Конечность*** – это свойство алгоритма, которое определяет завершение каждого действия в отдельности и алгоритма в целом за конечное число шагов.

Запишите свойства в тетрадь *(учащиеся записывают определения в тетрадь)*.

Действий не могут выполняться сами собой, у них всегда есть ***Исполнитель*** - это объект, умеющий выполнять определенный набор действий.

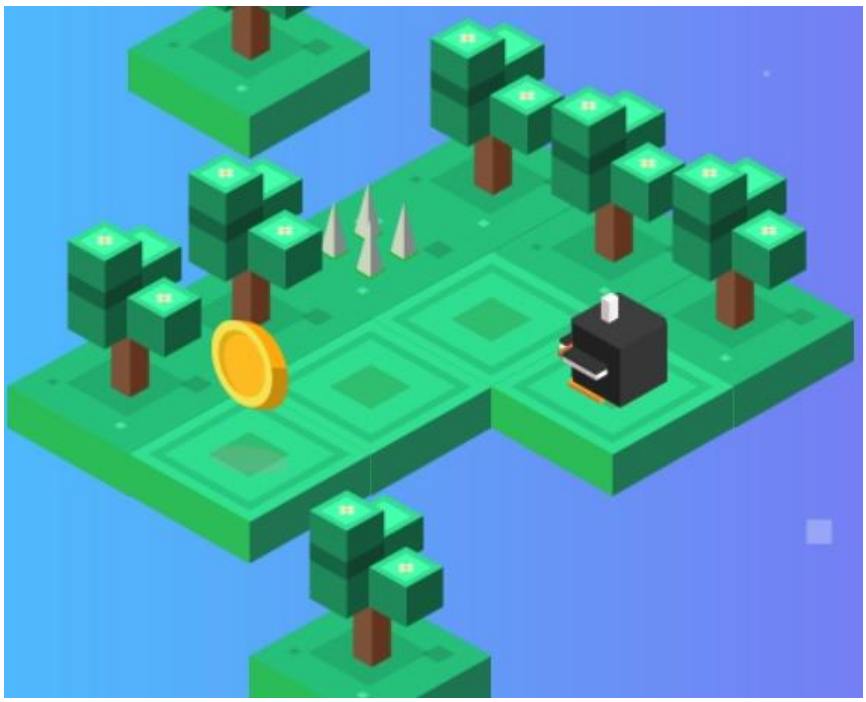
Исполнитель может делать только те команды, которые он знает, как выполнять, что называется ***системой команд.*** Для выполнения этих команд нужна определенная обстановка, в которой исполнитель может функционировать, что называется – ***средой исполнителя.***

Запишем определения в тетрадь *(учащиеся записывают определения в тетрадь)*.

Мы с вами на первом уроке Информатика познакомились с игрой ***«Algorithm City»***, скажите, а есть ли в этой игре Исполнитель? *(Ответ да, пингвин)*

А какие команды он знает? *(Ответ: Шаг вперед, поворот направо, налево, прыжок, взять, выполнение функций)*

Задание. Перед вами знакомая обстановка с игры ***«Algorithm City»***, на столах у вас есть небольшие карточки, заполните их, определите кто же является Исполнителем, впишите систему команд какие он знает и запишите шаги необходимые сделать персонажу для достижения результата (прохождение уровня)?



Выполняют работу на листках.

Историческая справка.

Великий узбекский математик и астроном аль-Хорезми (жившего в 9 веке), в своих трудах по арифметике и алгебре разработал правила выполнения четырёх арифметических операций над многозначными десятичными числами.

Эти правила определяют последовательность действий, которые необходимо выполнить, чтобы получить сумму чисел, произведение и т. д.

Первоначально только эти правила и назывались алгоритмами. В дальнейшем термин «алгоритм» стали использовать вообще для обозначения последовательности действий, приводящей к решению проблемы.

Научное определение понятие алгоритма дал А. Черч в 1930 году. Позже и другие математики вносили свои уточнения в это определение. В целом определение “Алгоритма” дают следующее: ***Алгоритм*** – описание последовательности действий (план), исполнение которых приводит к решению поставленной задачи за конечное число шагов.

Алгоритмизация – процесс разработки алгоритма (плана действий) для решения задачи.

Давайте запишем эти определения в тетрадь *(учащиеся записывают определения в тетрадь)*.

Виды алгоритмов

Линейный – описание действий, которые выполняются однократно в заданном порядке; Пример на слайде

Циклический – описание действий или группы действий, которые должны повторяться указанное число раз или пока не выполнено заданное условие. Пример на слайде

Разветвляющийся – алгоритм, в котором в зависимости от условия выполняется либо одна, либо другая последовательность действий. Пример на слайде.

В целом алгоритмы нужны для хранения и/или передачи, какого - либо плана действий или инструкций, исходя из нашего примера по запуску машины, вы уже знаете, что они могут быть как маленькими, которые удобно писать в виде текста или списка, так и очень большими, например инструкция для проверки гоночного болида может занимать десятки страниц текста. На изучение, которой уйдет очень много времени.

Поэтому Шаги можно описывать несколькими способами:

1.**Инструкция** (псевдокод, пронумерованные списки)

Формулы решения квадратных уравнений

1. Найдите дискриминант квадратного уравнения по формуле D=b^2-4*a*c
2. Если D>0, то пункт 3, иначе пункт 4
3. Квадратное уравнение имеет два корня. x=(-b±√D)/(2*a), перейдите к пункту 8
4. Если D=0, то пункт 5, иначе пункт 6
5. Квадратное уравнение имеет один корень. x=(-b/(2*a) , перейдите к пункту 8
6. Если D<0, то пункт 7, иначе пункт 8
7. Квадратное уравнение не имеет корней.
8. Ответ

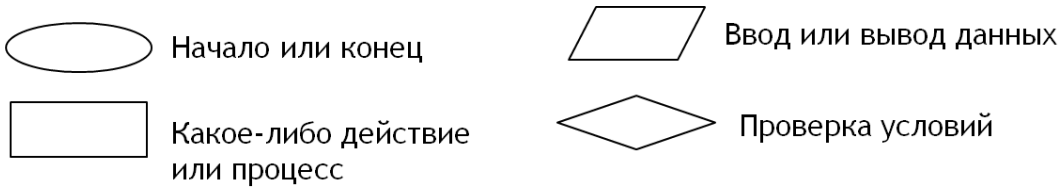
2.В виде **блок – схемы** (графический).

Часто пунктов много, и их последовательная запись сверху-вниз не даёт понять сам характер того, что есть. Воспринимать такую информацию в текстовом виде тяжело.

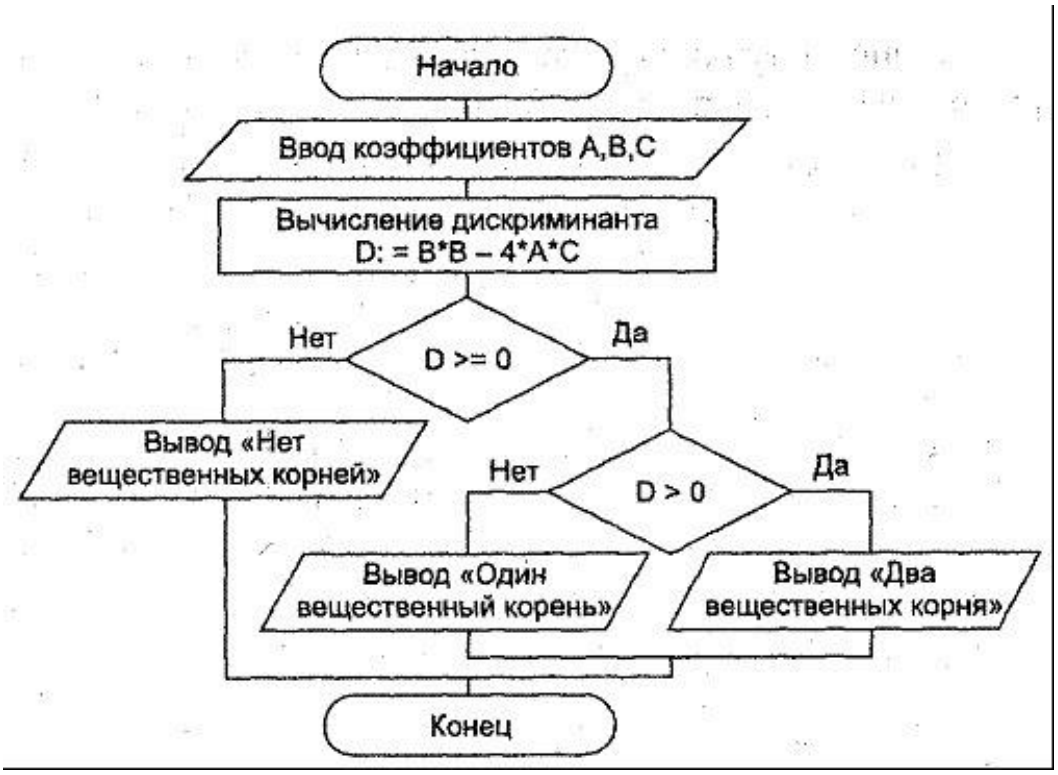
Вспомним наш пример, сложно держать всю инструкцию в голове, для ориентирования в ней и для перехода из пункта в пункт требуется время. А если будет инструкция к болиду?

Избавиться от длинных текстовых инструкций можно с помощью графического изображения действий, переходы между действиями обозначать стрелками.

Блок – схема - это схема алгоритма представляет собой систему связанных геометрических фигур. Каждая фигура означает один этап (шаг) процесса решения задачи и называется блоком. Порядок выполнения этапов указывается стрелками, соединяющими блоки.



Тогда инструкция по решению квадратных уравнений будет иметь вид:



Скажите, в каком виде проще воспринимать информацию? (на слайде два способа представления информации) *(Ответ: второй способ более наглядный).*

Если вы внимательно просмотрели данную блок-схему, то могли заметить, что дискриминант можно найти только для «*вещественных*

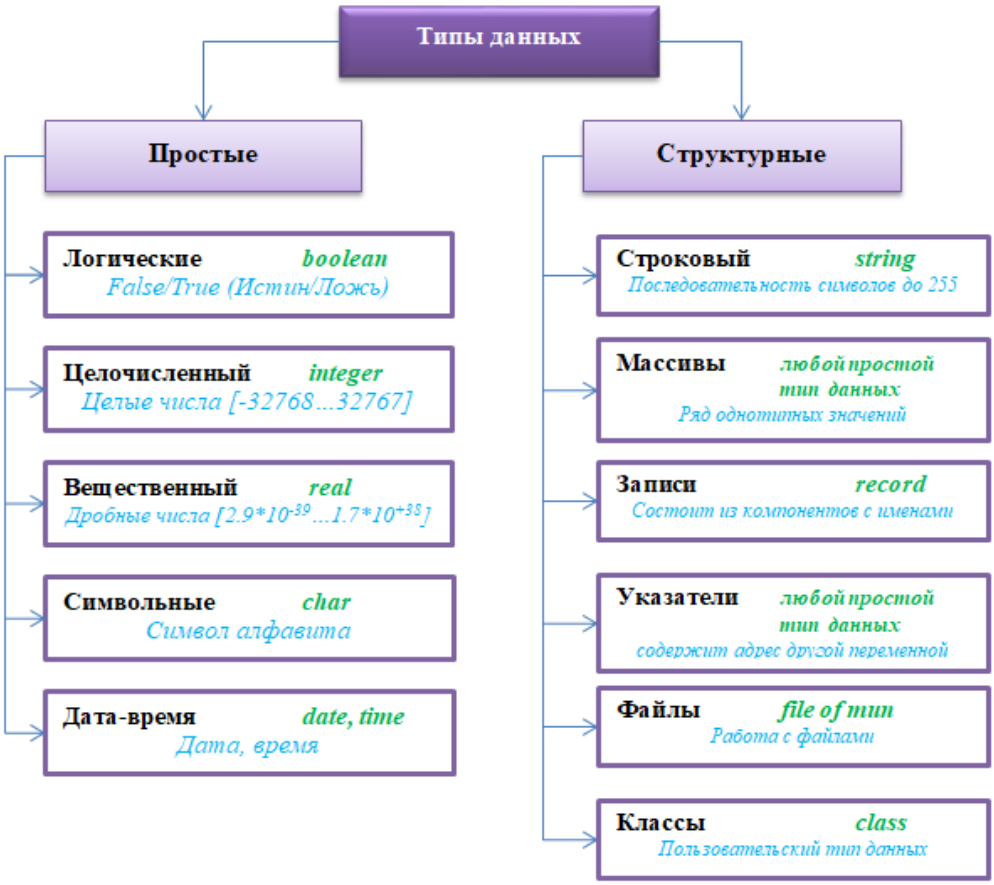
корней». А о чем идет речь, что такое «вещественное число»? *(Ответ: это дробные числа, числа с запятой).*

Типы данных

Информация это знаковая система, на прошлых уроках мы говорили что информация может представляться в разных видах: текст, числа, графика, звук. Когда же мы говорим о языке программирования, то подразумеваем под информацией строго определенное понятие. В данном случае информация — это данные и процедуры их обработки. Данные характеризуются своими типами, которые определяют:

- формат представления данных в памяти компьютера;
- область возможных значений;
- множество допустимых операций, применимых к данным.

В свою очередь типы данных делятся на простые и структурные



Давайте немного поговорим о типах данных, попробуем определить в приведенных примерах, какие лучше использовать типы данных?

- Если мы компьютере будем производить вычисления, по типу: a=2+b, и число b нам будет задавать сосед справа, то какие типы переменных мы будем задавать дя a и b?

(Возможный ответ: это зависит от того числа который нам скажет сосед, если скажет "7", то можно задать целое число, если "2,5", то вещественное)

- Если мы компьютеру передаем какое -либо сообщение, например слово "Привет", то какой тип данных будет?

(Возможный ответ: символьный)

Задание. Давайте преобразуем нашу первую инструкцию из текстового описания в графический вид. *Учащиеся выполняют задание.*

3. **Специальный язык**: когда шагов очень много, блок-схема становится снова не читаемой, понять которую сложно, и люди снова возвращаются к инструкциям, но записывают их уже на специальных языках, придуманных для (а) однозначной записи, (б) краткости записи, (в) выполнения машинами.

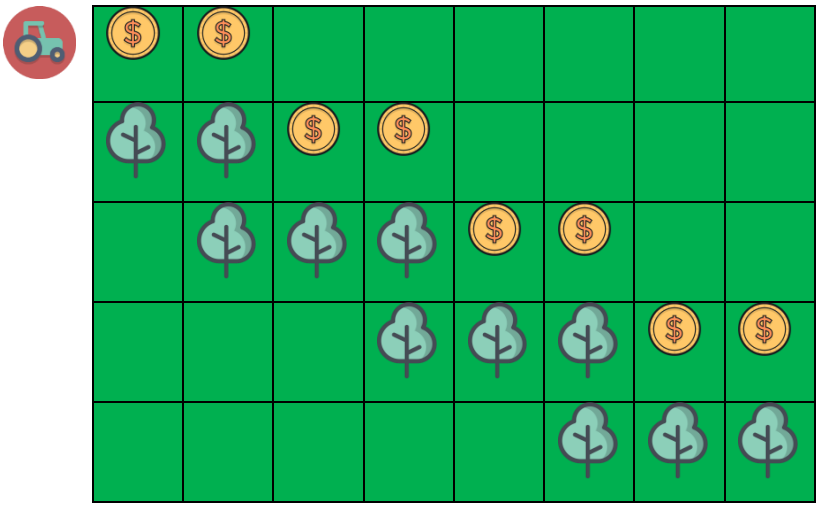
Пример: программа на паскале для решения квадратного уравнения.

```
BEGIN
    Writeln ('введите коэффициенты уравнения a,b,c');
    Readln(a,b,c);
    D:=b*b-4*a*c;
    If D>=0 then
        Begin
            X1:=(-b+sqrt(d))/(2*a);
            X2:=(-b-sqrt(d))/(2*a);
            Writeln('x1=',x1' x2=',x2);
        End;
    Else
        Writeln('действительных корней нет');
END.
```

Более подробно с Паскале мы познакомимся на практических занятиях, сейчас давайте попробуем порешать задачи в графическом виде.

4. Самостоятельная работа - 35 мин

Задание 1. Перед вами поле размером 8×5, в начале поля стоит трактор, ему необходимо собрать все монетки на поле. Напишите, какие действия должен сделать трактор, чтоб собрать все монетки? Составьте блок-схему.



Задание 2. Составьте словесное описание, блок-схему с указанием исходных, результирующих данных по каждой задаче.

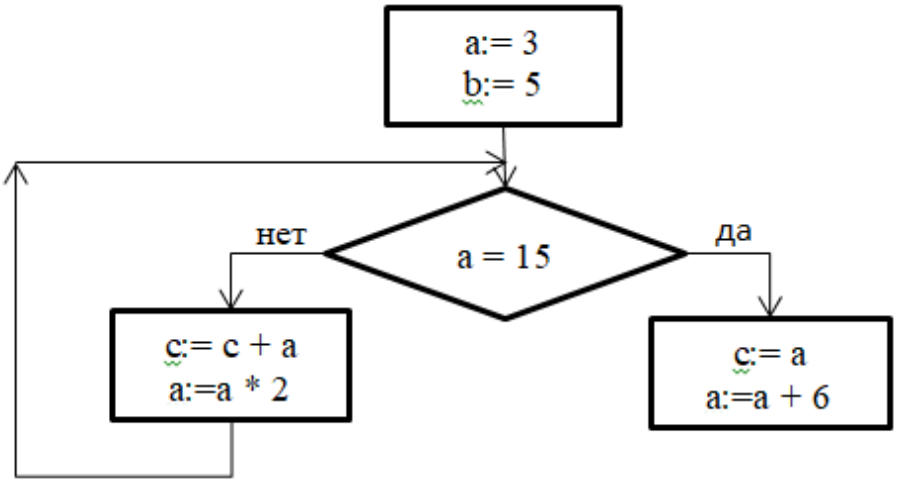
a) Автомобиль проехал три участка пути разной длины с разными скоростями. Найдите среднюю скорость автомобиля.

b) Расстояние между полицейской машиной и машиной преступника равно 240 м. Скорость полицейского автомобиля -40 м/с, машины преступника - 38 м/с. Как скоро полицейские догонят преступника?

c) В бригаде, работающей на уборке сена, имеется N сенокосилок. Первая сенокосилка работала m часов, а каждая следующая на 10 минут больше, чем предыдущая. Сколько часов проработала вся бригада?

Задание 3. По данным блок-схемам определите результат выполнения алгоритма.

а) Определите значение переменной *c* после выполнения фрагмента алгоритма.



5. Подведение итогов занятия - 2 мин.

Рефлексия: Что узнали нового сегодня? Какие трудности были?

(Сегодня на занятии вы узнали что такое алгоритм, какие бывают алгоритмы и как они изображаются графически. Также поговорили о типах данных.)

6. Домашнее задание - 2 мин.

§2.13. Повторение пройденного материала, доделать самостоятельную работу, задания прислать в Google Класс.

Код программы

```
uses GraphABC, ABCSprites, ABCObjects, Events, Timers;
const
    ROAD = 1;
    STONE = 2;
    VOID = 0;
    HERO_NORTH = 0;
    HERO_EAST = 1;
    HERO_SOUTH = 2;
    HERO_WEST = 3;
    TILE_DIAG = 81;
    TILE_X0 = -100;
    TILE_Y0 = 500;
    COIN_SIZE = 30;

var
  sprite_hero : SpriteABC;
  t   : TextABC;
  p1, p2      : PictureABC;
  coins_data  : array[1..20, 1..20] of integer;
  coins_pict  : array[1..20, 1..20] of PictureABC;
  map_data    : array[1..20, 1..20] of integer;
  map_pict    : array[1..20, 1..20] of PictureABC;
  hero_i, hero_j, hero_dir : integer;

procedure LoadMap;
const map_filename = 'map.txt';
var   i, j      : integer;
    map_file  : file;
```

```
    c        : char;
    line     : string;
begin
  Assign(map_file, map_filename);
  Reset(map_file);
  for j := 20 downto 1 do
  begin
    for i := 1 to 20 do
    begin
      read(map_file, c);
      coins_data[i, j] := 0;
      case c of
        '.': map_data[i, j] := VOID;
        's': map_data[i, j] := STONE;
        'r': map_data[i, j] := ROAD;
        'c': begin
               map_data[i, j] := ROAD;
               coins_data[i, j] := 1;
             end;
        'H': begin
               map_data[i, j] := ROAD;
               hero_i := i;
               hero_j := j;
               hero_dir := HERO_NORTH;
             end;
      end;
    end;
    read(map_file, c, c);
  end;
end;

function scr_x(i, j : integer) : integer;
```

```
var
  dx, dy : integer;
begin
  dx := TILE_DIAG div 2;
  dy := dx div 2;
  scr_x := i * dx + j * dx + TILE_X0;
end;

function scr_y(i, j : integer) : integer;
var
  dx, dy : integer;
begin
  dx := TILE_DIAG div 2;
  dy := dx div 2;
  scr_y := i * dy - j * dy + TILE_Y0;
end;

procedure DrawMap;
var   i, j, dx, dy   : integer;
begin
  dx := TILE_DIAG div 2;
  dy := dx div 2;
  for i := 1 to 20 do
  for j := 20 downto 1 do
  begin
    case map_data[i, j] of
      STONE: map_pict[i, j] := new PictureABC(scr_x(i, j), scr_y(i, j),
'tile_stone.png');
      ROAD: map_pict[i, j] := new PictureABC(scr_x(i, j), scr_y(i, j),
'tile_road.png');
    end;
    if coins_data[i, j] = 1 then
```

```
      coins_pict[i, j] := new PictureABC(scr_x(i, j) + TILE_DIAG div 2 -
COIN_SIZE div 2,      scr_y(i, j),      'heart-coins.spinf');
  end;
  sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_N.png');
end;

procedure hero_take_coin;
begin
  if coins_data[hero_i, hero_j] <> 0 then
  begin
    coins_data[hero_i, hero_j] := 0;
    coins_pict[hero_i, hero_j].Destroy;
  end
  else
    writeln('ERROR!');
end;

procedure hero_forward;
begin
  case hero_dir of
    HERO_NORTH: inc(hero_j);
    HERO_SOUTH: dec(hero_j);
    HERO_EAST:  inc(hero_i);
    HERO_WEST:  dec(hero_i);
  end;
  sprite_hero.MoveTo(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2);
end;

procedure hero_turn_right;
begin
```

```
  sprite_hero.Destroy;
  case hero_dir of
    HERO_NORTH:
    begin
      hero_dir := HERO_EAST;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_E.png');
    end;

    HERO_SOUTH:
    begin
      hero_dir := HERO_WEST;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_W.png');
    end;

    HERO_EAST:
    begin
      hero_dir := HERO_SOUTH;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_S.png');
    end;

    HERO_WEST:
    begin
      hero_dir := HERO_NORTH;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_N.png');
    end;
  end;
end;
```

```
procedure hero_turn_left;
begin
  sprite_hero.Destroy;
  case hero_dir of

HERO_NORTH:
    begin
      hero_dir := HERO_WEST;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_W.png');
    end;

    HERO_EAST:
    begin
      hero_dir := HERO_NORTH;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_N.png');
    end;

    HERO_SOUTH:
    begin
      hero_dir := HERO_EAST;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_E.png');
    end;

    HERO_WEST:
    begin
      hero_dir := HERO_SOUTH;
      sprite_hero := new SpriteABC(scr_x(hero_i, hero_j), scr_y(hero_i, hero_j) -
TILE_DIAG div 2, 'hero_S.png');
    end;
```

```
  end;
end;

begin
  SetWindowSize(1200, 900);
  writeln('Loading map..');
  LoadMap;
  DrawMap;
  Window.Clear(clGreen);

  hero_forward;
  hero_forward;
  hero_forward;
  hero_take_coin;
end.
```