МИНИСТЕРСТВО ОБРАЗОВАНИЯ И НАУКИ РФ
федеральное государственное бюджетное образовательное учреждение высшего образования
КРАСНОЯРСКИЙ ГОСУДАРСТВЕННЫЙ ПЕДАГОГИЧЕСКИЙ УНИВЕРСИТЕТ
им. В.П.АСТАФЬЕВА
(КГПУ им. В.П.Астафьева)

Институт/факультет Институт математики, физики и информатики
(полное наименование института/факультета/филиала)

Выпускающая(ие) кафедра(ы) Базовая кафедра информатики и информационных технологий в образовании .
(полное наименование кафедры)

Еричук Максим Сергеевич

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА

Тема: Обучение программированию в основной школе с использованием средств визуализации.

Направление подготовки 44.03.01 Педагогическое образование
Направленность (профиль) образовательной программы Информатика

ДОПУСКАЮ К ЗАЩИТЕ

	Зав.кафедрой	д. п. н., профессор, Пак Н.И. (ученая степень, ученое звание, фамилия, инициалы)
		(дата, подпись)
	Руководитель	к.п.н., Кулакова И.А. (ученая степень, ученое звание, фамилия, инициалы)
	Дата защиты	
	Обучающийся	Еричук Максим Сергеевич (фамилия, инициалы)
		(дата, подпись)
	Оценка	
		(прописью)

Красноярск, 2017

Содержание

Введение

Согласно исследованиям Росстата, самой востребованной профессией в ближайшие годы будет профессия программиста. Подготовка программистов начинается со школьного курса информатики, где учащиеся впервые знакомятся с различными языками программирования.

Среди множества языков программирования большую популярность у учителей информатики имеет Паскаль. Так как тема «Программирование» является одной из самых трудных для изучения в курсе «Информатика и ИКТ» в школе, для начинающего программиста увидеть результат своего труда в виде напечатанного на консоли числа представляется не очень наглядным и интересным. Решение только вычислительных задач формирует у обучающих мнение о программировании как о скучном и неинтересном предмете.

Использование графических возможностей языков программирования помогает при изучении данного материала, т.к. позволяет:

1) сделать изучение программирования эмоционально привлекательным;

2) наглядно продемонстрировать результат выполнения алгоритма;

3) расширить спектр решаемых задач.

Все выше сказанное обуславливает **актуальность** данной работы.

В различных учебниках тема «Алгоритмизация и программирование» не рассмотрена достаточно полно, глубоко и доступно, и предполагается разное количество часов на ее изучение данной темы. Поэтому задача учителя в школе состоит в разработке такой методики, которая максимально упростит развитие способности программировать, что очень важно для

большинства людей в современном техническом мире. Учителю приходится пользоваться личными разработками уроков, использовать ранее

наработанный опыт и учебники, которые прямо или косвенно содержат материал для изучения выбранного языка программирования.

Найти необходимый материал для обучения можно и в сети Интернет, однако какой-либо методической системы обучения программированию основанной на использовании средств визуализации нет, что определяет **проблему** исследования.

Объект исследования: процесс обучения программированию на уроках информатики в основной школе.

Предмет исследования: методика обучения программированию с использованием графических возможностей Паскаля.

Цель: Разработка дидактических материалов для обучения программированию на Паскале с использованием средств визуализации.

Задачи:

1. Раскрыть теоретические аспекты обучения программированию в основной школе.
2. Изучить возможности обучения программированию с использованием средств визуализации.
3. Разработать систему задач по программированию на Паскале с использованием графических возможностей.
4. Составить рекомендации по использованию средств визуализации в обучении школьников программированию

Работа состоит из введения, двух глав, заключения, списка литературы и приложений.

Глава 1. Теоретические аспекты обучения программированию в школе

1.1 Обучение программированию в основной школе

Информатика представляет собой комплексную научную и инженерную дисциплину, которая изучает все аспекты разработки, проектирования, создания, оценки, функционирования машинизированных систем переработки информации, их применения и воздействия на раз личные области социальной практики [4, С. 17].

Часть информатики, обслуживающая проблемы средней школы, получила название школьной информатики. Впервые в отечественной литературе этот термин был озвучен А.П. Ершовым. Им школьная информатика определялась как ветвь информатики, занимающаяся исследованием и разработкой программного, технического, учебно-методического и организационного обеспечения применения ЭВМ в школьном учебном процессе.

Изучение информатики и информационно-коммуникационных технологий направлено на достижение следующих целей:

– освоение системы базовых знаний, отражающих вклад информатики в формирование современной научной картины мира, роль информационных процессов в обществе, биологических и технических системах;

– овладение умениями применять, анализировать, преобразовывать информационные модели реальных объектов и процессов, используя при этом информационные и коммуникационные технологии, в том числе при изучении других школьных дисциплин;

– развитие познавательных интересов, интеллектуальных и творческих способностей путем освоения и использования методов информатики и средств ИКТ при изучении различных учебных предметов;

– воспитание ответственного отношения к соблюдению этических и правовых норм информационной деятельности;

– приобретение опыта использования информационных технологий в индивидуальной и коллективной учебной и познавательной, в том числе проектной деятельности [8, С. 11].

Преподавание информатики в средней школе выполняет следующие функции:

1. Образовательная функция заключается в организации процесса обучения, способствующем становлению человека как субъекта активности, овладению школьниками системой знаний, дающей представление о предмете информатики, ее методах и приложениях. Она предполагает необходимость выделения понятий, осуществляющих взаимосвязь с другими науками, важность формирования определенной системы взглядов на окружающий мир, умение решать задачи прикладной направленности.

2. Развивающая функция заключается в формировании у учащихся познавательных психических процессов и свойств личности: внимания, памяти, мышления, познавательной активности и самостоятельности, способностей. К развивающей функции обучения относится формирование логических приемов мыслительной деятельности (анализ, синтез, обобщение, абстрагирование и т.п.), общеучебных приемов. Развивающая функция предполагает ориентацию на выявление и реализацию в процессе обучения потенциальных возможностей информатики

3. Воспитательная функция. Реализация воспитательной функции обучения предполагает его ориентацию на формирование интеллектуальных и морально-этических компонентов личности, качеств мышления, характерных для информационной деятельности. К воспитательной функции относится формирование интереса к изучению информатики, развитие устойчивой мотивации к учебной деятельности.

4. Профориентационная функция. Уже на базовом этапе в рамках предпрофильной подготовки курс информатики должен давать учащимся сведения о профессиях, связанных с ЭВМ и информатикой, с различными

приложениями изучаемых в школе дисциплин, опирающимися на использование ЭВМ. Важен и «бытовой» аспект – готовность молодых людей к грамотному использованию компьютерной техники и других средств информационных и коммуникационных технологий в быту и в повседневной жизни.

5. Эвристическая функция предполагает создание в процессе обучения условий, обеспечивающих развитие способностей ребенка. Необходимо создание учителем на уроке и вне его среды, благоприятной для развития личности, обеспечение самореализации личностного потенциала и побуждения к поиску собственных, лично значимых результатов в обучении. К этой же функции отнесем усвоение эвристических приемов и методов познания, их реализацию на практике.

6. Прогностическая функция обучения информатике обусловлена во многом усилением эвристической и развивающей функций, которые предусматривают включение школьника в процесс открытия фактов, их обоснования, анализа различных способов аргументации.

7. Эстетическая функция. Информатика обладает значительным эстетическим потенциалом, который должен использоваться для приобщения школьников к красоте, воспитания у них эстетических вкусов и переживаний, в том числе за счет курсов интегративного характера, связанных с Web-дизайном, компьютерной графикой и анимацией, обработкой звука и видео, разработкой мультимедийных средств и т.д.

8. Контрольно-оценочная функция заключается в необходимости осуществления контроля, коррекции и оценки знаний и умений учащихся. Методика обучения информатике ищет новые формы контроля усвоения учебного материала. Наряду с традиционным опросом учащихся, уроками-зачетами, уроками коррекции знаний и т.д., все большее значение приобретает тестирование, особенно в связи с введением ЕГЭ.

9. Информационная функция состоит в том, что в процессе обучения ученик знакомится с историей возникновения идей, их развитием, биографией ученых, разными точками зрения на те или иные концепции, борьбой ученых за утверждение научных взглядов, а также с различными приложениями информатики и открытиями в области информатики.

10. Корректирующая функция заключается в корректировании информации, получаемой учащимися. Ученик получает информацию из многих источников внешней информационной среды. Значение и сущность информации, полученной из различных источников, может быть весьма неоднозначно как с научной, так и этической, моральной и т.д. точек зрения. Зная конкретные ситуации, учитель должен откорректировать информацию, помочь ученику разобраться в ней и правильно ее оценить.

11. Интегрирующая функция. Ее сущность заключается в формировании системности знаний, в понимании взаимосвязи между изучаемыми понятиями, теоремами, способами деятельности, методами, в иерархии между отдельными видами знаний, в умении применять различные методы в решении задач, в выделении межпредметных связей, в понимании роли информатики в науке, технике и жизнедеятельности общества.

Во второй половине 90-х годов в РФ были введены понятия базового и профильного курсов информатики. Базовый курс, предназначенный обеспечить государственный образовательный стандарт по информатике, был разделен на следующие тематические разделы:

- информации и информационных процессов;
- представления информации;
- аппаратной части компьютера;
- формализации и моделирования;
- алгоритмизации и программирования;
- информационных технологий [9, С. 24].

В настоящее время изучение темы «Алгоритмизация программирования» в школе проводится для разных категорий учащихся, главным образом это связано с появлением профильного обучения. Одна категория учащихся изучает программирование в соответствии с содержанием образования по информатике, обязательного для всех учащихся общеобразовательных школ. Для основной школы цель обучения определяется изучением основных алгоритмических конструкций с использованием простых типов данных и массивов. Другая категория учащихся изучает программирование на углубленном уровне, что соответствует обучению в таких профильных классах, как информационно-коммуникационное и физико-математическое направления.

Курс информатики по ФГОС обязательно:

· в 5 – 6 классах, по усмотрению школы – от 17 до 34 учебных часов в год из расчета 0,5 - 1 учебных часов в неделю.

· в 7 – 8 классе, 34 учебных часов в год из расчета 1 учебный час в неделю;

· в 9 классе, 68 учебных часов в год из расчета 2 учебных часа в неделю;

Одной из дидактических задач образовательного учреждения является интеллектуальное развитие учащегося, важной составляющей которого является алгоритмическое мышление.

В итоге изучения учащиеся должны знать:

– что такое алгоритм, какова роль алгоритма в системах управления;

– в чем состоят основные свойства алгоритма;

– способы записи алгоритмов: блок-схемы, учебный алгоритмический язык;

– основные алгоритмические конструкции: следование, ветвление, множественный выбор, цикл, структуры алгоритмов;

– назначение вспомогательных алгоритмов, технологии построения сложных алгоритмов: метод последовательной детализации и сборочный (библиотечный) метод;

– основные свойства величин в алгоритмах обработки информации: что такое имя, тип, значение величины; смысл присваивания;

– назначение языков программирования;

– в чем различие между языками программирования высокого уровня и машинно-ориентированными языками;

– правила представления данных на одном из языков программирования высокого уровня (например, на языке Pascal);

– правила записи основных операторов: ввода, вывода, присваивания, цикла, ветвления;

– правила записи программы;

– что такое трансляция;

– назначение систем программирования;

– содержание этапов разработки программы: алгоритмизация, кодирование, отладка и тестирование [8, С. 24].

При этом учащиеся должны уметь:

– пользоваться языком блок-схем, понимать описания алгоритмов на учебном алгоритмическом языке;

– выполнять трассировку алгоритма для известного исполнителя;

– составлять несложные линейные, ветвящиеся и циклические алгоритмы управления на одном из учебных исполнителей;

– выделять подзадачи, определять и использовать вспомогательные алгоритмы;

– составлять несложные программы решения вычислительных задач с числами;

– программировать простой диалог;

– работать в среде одной из систем программирования;

– осуществлять отладку и тестирование программы [8, С. 27].

Изучение программирования в школьном курсе информатики подчиняется ряду принципов:

1. Принцип многоуровневости. Этот принцип связан с тем, что в алгоритмизации и программировании, как ни в одной другой области деятельности, все темы, элементы языка тесно переплетены между собой. Поэтому некоторую трудность представляет определение порядка изучения понятий и конструкций языка, приемов и методов программирования. Большинство учебников построены таким образом, что сначала рассматриваются элементы выбранного языка программирования (константы и переменные, все их типы, все операции над ними, правила построения выражений и т. д.), а потом последовательно изучаются операторы, методы и технологии программирования. При этом, как правило, сразу в одном пункте приводятся все возможности изучаемого элемента языка, например, оператора, хотя некоторые из них редко используются или достаточно сложные. Аналогично при изучении какой-нибудь другой системы (не обязательно системы программирования) приводится полное описание некоторого элемента, хотя на начальном этапе достаточно знать ее простейшую форму, лишь некоторые ее возможности.

1. Принцип перехода от частного к общему. Языки программирования рекомендуется изучать по следующей схеме: наглядная простая задача, которая требует для своего решения введения нового элемента, т.е. показывается необходимость изучаемого элемента; пример использования нового элемента при решении этой задачи; общий вид конструкции, ее

формальное описание; примеры на закрепление; выполнение индивидуальных заданий на ЭВМ и (или) упражнений без использования компьютеров.

2. Принципы сравнения и повторения. При изучении многих тем педагоги руководствуются принципом сравнения, который предполагает анализ различных алгоритмов и (или) программ решения одной и той же задачи, выбор из них наилучшего. Этот же принцип используют также для того, чтобы показать необходимость той или иной конструкции языка, быстрее и лучше понять ее. Для этого записывают несколько вариантов некоторого фрагмента программы решения одной и той же задачи, используя разные элементы.

3. Принцип индивидуальных заданий. При обучении программированию следует исходить из того, что развить алгоритмическое мышление школьника, научить его программировать и решать задачи с применением ЭВМ можно только при условии, если каждый из них будет регулярно составлять программы, выполнит самостоятельно несколько индивидуальных заданий. Каждое такое задание закрепляет и проверяет знания по одной или нескольким наиболее важным темам и предполагает выполнение следующих этапов: изучение постановки задачи; разработку алгоритма ее решения; запись алгоритма в виде программы; подготовку программы на компьютере в соответствующей системе программирования; отладку и тестирование программы; анализ результатов и подготовку отчета. В отличие от традиционной методики, когда всему классу предлагается общая задача, при такой организации работы школьника исключается формальное переписывание друг у друга. При этом можно учесть способности школьников, варьируя уровнем сложности заданий. Важно также что при выполнении индивидуальных заданий повышается активность на занятиях и ответственность за его своевременное выполнение. При этом

эффективность обучения возрастает, если использовать следующие методические приемы:

– выполнение одного задания вдвоем (наиболее способный ученик со слабым);

– доклад наиболее интересных, оригинально, нестандартно выполненных заданий на семинарском занятии и его обсуждение;

– обмен готовыми программами друг с другом с объяснением;

– конкурс на лучшую программу решения одной и той же задачи; – бригадный метод работы (одно задание на 2-4 человека). Это позволяет ученикам обмениваться идеями друг с другом, искать вместе ошибки, учит работать в коллективе [9, С.12].

Организационные формы и средства обучения по теме алгоритмизации и программирования используются, как и на любом другом уроке. Обычно в начале изучения новых команд и операторов преимущество отдают фронтальным видам организации урока. При этом используют плакаты, кодоскоп, эпипроектор и пр. Закрепление материала проходит в виде групповой деятельности или индивидуально, а контрольные задания учащиеся, как обычно, выполняют самостоятельно. На практике преимущественно урок строят по следующей схеме:

– 10 мин. фронтальная работа с классом (проверка домашнего задания, выполнение устных упражнений);

– 15 мин. объяснение нового материала;

– 20 мин. работа за компьютером, выполнение заданий.

Основным средством изучения программирования в школьном курсе информатики является задача.

Все задачи по программированию можно разделить на два больших класса: задачи- программы и упражнения. Первый из них включает задачи на

составление алгоритма и полной программы на выбранном языке и их анализ, а в машинном варианте, кроме этого, и её отладку. В упражнениях требуется записать один или несколько вариантов элемента языка (записать выражение, оператор) или части программы и (или) проанализировать их. При анализе программы (ее части) надо ответить на ряд вопросов, например, есть ли ошибки, всегда ли ошибка будет проявляться, как повлияет на результат какое-нибудь изменение в программе и т.п.

Первый класс задач является более важным, и его необходимо чаще практиковать, особенно для закрепления одной или нескольких тем и при выполнении индивидуальных заданий. Но на начальном этапе изучения темы и при объяснении наиболее сложных принципиальных вопросов нельзя игнорировать упражнения, которые играют подготовительную, вспомогательную роль.

Упражнения особенно полезны при работе с отстающими школьниками. Частным случаем упражнений являются тесты. Одним из этапов обучения программированию, важным условием умения составлять качественные программы является понимание готовых программ, умение их «читать». Поэтому полезно выполнять упражнения на восстановление постановки задачи, для которой записана программа или ее часть. В случае затруднения можно рекомендовать исполнить ее для одного или нескольких тестов, или составить блок-схему. Необходимо обратить внимание на такие упражнения, в которых надо сравнить несколько вариантов программы или алгоритма. При этом необходимо выбрать правильный вариант из предложенных. Полезны упражнения, в которых требуется исследовать на эффективность программу или её часть, выбрать наилучший из правильных вариантов или записать, если можно, более эффективный вариант.

При изучении основных типов алгоритмов, наиболее важных и сложных операторов и приемов программирования полезно использовать задачи и упражнения на составление и анализ блок-схем. Кроме простоты и

наглядности, важной отличительной их особенностью и достоинством является то, что они не зависят от языка программирования. Научившись записывать алгоритмы в виде блок-схем, достаточно освоить синтаксические правила конкретного языка, чтобы можно было разрабатывать эффективные программы.

Блок-схемы способствуют быстрому развитию алгоритмического мышления. Их можно использовать как в машинном, так и в без машинном вариантах изучения программирования. К сожалению, в последнее время в учебниках по программированию этому удаляется недостаточно внимания. Блок-схемы можно изучать как параллельно с языком программирования, так и последовательно. В первом случае сначала в течение 1-2 уроков учащимся дают простейшие элементы блок-схем, правила их разработки и требования к ним.

При изучении методов программирования и наиболее сложных операторов языка составляется при необходимости блок- схема алгоритма, а потом с ее помощью разрабатывается программа. При таком способе, хотя и требуется больше времени, школьники лучше понимают динамический смысл выполнения программ. Последовательное изучение блок-схем и языка программирования заключается в том, что на начальном этапе изучения программирования несколько уроков учащиеся разрабатывают и анализируют блок-схемы различных типов алгоритмов (линейных, разветвляющихся, циклических с известным и неизвестным количеством повторений, итерационных, алгоритмы с вложенными циклами). Затем эти же типы алгоритмов изучаются на уровне программ.

1.2. Изучение графических возможностей языка программирования Паскаль в школе

Программирование – это наиболее важный раздел курса «Информатика и ИКТ», изучение которого позволяет решать целый ряд дидактических и педагогических задач.

С помощью языков программирования можно создавать свои программы, решать нестандартные задачи. Программирование вырабатывает у учащихся четкое логическое мышление, аккуратность и внимательность, развивает находчивость, изобретательность, фантазию и творческие способности.

Ядро системы программирования составляет язык. Язык программирования – это формальная знаковая система, которую понимает компьютер. Язык программирования определяет набор лексических, синтаксических и семантических правил, используемых при написании алгоритмов компьютерных программ. Язык программирования предназначен для того, чтобы компьютер понимал инструкции по выполнению той или иной программы, написанной на соответствующем языке программирования.

Выбор языка и системы программирования имеет принципиальное значение, т.к. от этого во многом зависит методика изучения курса, содержание и последовательность предъявления учебного материала, система учебных заданий и, главное, вся дальнейшая работа по овладению программированием для решения реальных практических задач на компьютере [13].

Языки программирования в школьном курсе информатики делятся на две группы: учебные языки (КуМир, Робик, Рапира, АЯ) и профессиональные языки (BASIC, Pascal, Си, Visual Basic, C++…), профессиональные в свою очередь делятся на процедурные и объектно- ориентированные.

Учебный язык программирования — язык программирования, предназначенный для обучения. В качестве таковых разрабатывались такие языки как BASIC и Паскаль. Из разработанного для обучения языка ABC вырос Python. Популярным языком, разработанным специально для образования, является LOGO. Специально для российских школ разработана языковая среда КуМир. Набирает популярность созданный в Массачусетском технологическом институте язык визуального программирования Scratch и тому подобные среды программирования. Учебный язык должен обеспечивать простоту, ясность и удобочитаемость конструкций. Излишняя гибкость, «вседозволенность» синтаксиса может затруднить понимание программ. С этим связаны преимущества использования в образовательном процессе языков семейства Pascal перед Си-подобными языками.

При выборе языка программирования важны такие факторы, как его новизна, эффективность реализации (в виде компилятора или интерпретатора). Фактор распространённости имеет как психологическое значение (влияя на мотивацию учащихся), так и практическое (востребованность получаемых знаний без необходимости переучивания). Учебный язык программирования должен обеспечивать плавный переход от псевдокода к собственно программированию. Полезным в обучении может быть возможность использования национальной лексики для ключевых слов и идентификаторов. Альтернативой относительно трудоёмким для изучения комплексным языкам программирования общего назначения могут составить простые мини-языки, в которых, для наглядности, имеется графический исполнитель, вроде черепашки в Лого — первом и одном из самых известных таких языков.

Профессиональные языки программирования, изучаемые в школе, представлены в таблице 1.

Таблица 1. Профессиональные языки программирования

BASIC	Язык подходит на начально этапе изучения программирования в основной школе, развивает алгоритмическое и логическое мышление.
Pascal	Язык ориентирован на структурную методику программирования. Особенностями языка являются строгая типизация и наличие средств структурного программирования.
Си	Язык программирования создавался с целью упростить написание больших программ с минимизацией количества ошибок по правилам процедурного программирования.
Visual Basic	Язык Visual Basic унаследовал стиль и отчасти синтаксис своего предка — языка Бейсик. В то же время Visual Basic сочетает в себе процедуры и элементы объектно- ориентированных и компонентно-ориентированных языков программирования. Среда разработки VB включает инструменты для визуального конструирования пользовательского интерфейса. Язык подходит как для обучения структурному программированию в основной школе, так и для обучения программированию на профильном уровне, так как развивает объектно-ориентированный стиль мышления. Подходит для выполнения практических работ и практикумов, предусмотренных в стандарте.
C++	Поддерживает такие парадигмы программирования как процедурное программирование, объектно- ориентированное программирование, 23 обобщённое программирование, обеспечивает модульность, раздельную компиляцию, обработку исключений, абстракцию данных, объявление типов (классов) объектов, виртуальные функции. Стандартная библиотека включает, в том числе, общеупотребительные контейнеры и алгоритмы. C++ сочетает свойства как высокоуровневых, так низкоуровневых языков. В сравнении с его предшественником — языком C, — наибольшее внимание уделено поддержке объектно-ориентированного и обобщённого программирования. Язык подходит для обучения программированию на профильном уровне, так как развивает объектно-ориентированный стиль мышления. Подходит для выполнения практических работ и практикумов, предусмотренных в стандарте.

Паскаль или Pascal – процедурный язык программирования, часто использующийся для обучения структурному программированию. Такие особенности как строгая типизация, наличие средств структурного программирования, а также понятный и простой синтаксис в сочетании с минимальными синтаксическими неоднозначностями, делают этот язык удобной платформой для освоения навыков алгоритмизации и понимания принципа работы программ.

К изучению языка Паскаль учащиеся приступают в 9 классе, и осваивают следующие темы:

типы данных, используемых в языке Паскаль, структура программы, оператор присваивания, организация ввода и вывода данных, ввод данных с клавиатуры, числовые типы данных (целочисленный тип данных), символьный и строковый типы данных, логический тип данных, программирование линейных и разветвляющихся алгоритмов, условный оператор, составной оператор, программирование циклических алгоритмов, программирование циклов с заданным условием продолжения работы, программирование циклов с заданным условием окончания работы, программирование циклов с заданным числом повторений, одномерные массивы целых чисел, описание массива, заполнение массива, вывод массива, процедуры, функции. [11]

Таким образом, графика в школе остается незатронутой. Однако, включение графики в изучение языка Паскаль и последующее ее применение для освоения дальнейших тем помогает реализовывать следующие принципы обучения:

Принцип наглядности – учащиеся будут сразу видеть результат своих действий, за счет чего возникает лучшее понимание назначения операций и работы программы. Это положительно влияет на долгосрочность усвоения материала, развивает склонность к критическому мышлению, повышает сознательность и активность.

Принцип активности – как правило, тема графика является одной из любимых тем учащихся за счет ее творческого характера, что в свою очередь приводит к повышению познавательного интереса и мотивации и, как следствие, активности учащихся.

Принцип сознательности – результат работы с графикой крайне нагляден, поэтому учащиеся четко представляют, что от них требуется, в следствие чего учащиеся ясно осознают цель своих действий.

В состав языка Паскаль входят буквы русского алфавита, но они могут использоваться только при вводе и выводе данных строкового типа.

Начиная с версии 4.0, в состав Turbo Pascal включена мощная библиотека графических программ Graph, остающаяся практически неизменной во всех последующих версиях.

Модуль Graph содержит обширный набор типов, констант, процедур и функций для управления графическим режимом работы экрана. С помощью подпрограмм, входящих в модуль Graph, можно создавать разнообразные графические изображения и выводить на экран текстовые надписи стандартными или разработанными программистом шрифтами. Подпрограммы модуля Graph после соответствующей настройки могут поддерживать различные типы аппаратных графических средств. Настройка на имеющиеся в распоряжении программиста технические средства графики осуществляется специальными программами – драйверами. Драйвер хранится в отдельном файле на диске и содержит как исполняемый код, так и необходимые ему для работы данные.

Модуль Graph содержит обширный набор процедур и функций, позволяющий управлять графическим режимом работы экрана: создавать разнооборазные графические изображения и выводить на экран текстовые надписи. Как уже говорилось раннее, особенностями языка Pascal, является строгая типизация и наличие средств структурного (процедурного) программирования. Язык Pascal относительно прост в изучении, довольно ясен и логичен и, будучи первым изучаемым языком программирования, приучает к хорошему стилю, воспитывает дисциплину структурного программирования.

PascalABC.NET — свободно распространяющийся компилятор языка Object Pascal для .NET, предназначенный для обучения современному программированию. Разработка ведется коллективом кафедры алгебры и дискретной математики факультета математики, механики и компьютерных наук ЮФУ.

PascalABC.NET является развитием проекта Pascal ABC;. Pascal ABC разработан в 2002 году сотрудниками факультета математики, механики и компьютерных наук Южного федерального университета (Ростов-на-Дону, Россия) во главе с С.С. Михалковичем. Целью авторов было создание обучающей среды программирования, более современной, чем Borland Pascal и Turbo Pascal, более простой для изучения, чем Borland Delphi, но в то же время близкой к стандартным компиляторам языка.

Интерпретатор Pascal ABC разработан в среде Delphi для Win32 и реализует язык, примерно соответствующий Object Pascal. Ряд возможностей исходного языка признаны ненужными для обучения и не реализованы. Некоторые языковые конструкции (например, модули и методы) могут использоваться в упрощенном виде на ранних этапах обучения. Все это позволяет максимально упростить переход от простейших структурных программ к модульному и объектно-ориентированному программированию.

Система Pascal ABC позволяет:

- работать с графикой.
- создавать событийные приложения.
- работать с исполнителями Робот и Чертежник.
- выполнять проверяемые задания, генерирующие случайные входные данные для задач и проверяющие правильность ответа. Для этого используется электронный задачника Programming Taskbook, содержащий 200 учебных заданий по следующим темам:
- скалярные типы данных и управляющие операторы;
- обработка последовательностей;
- минимум и максимум;
- одномерные и двумерные массивы;
- символы и строки;

- типизированные и текстовые файлы;
- процедуры и функции, рекурсия;
- указатели и динамические структуры данных.

Данный компилятор выбран нами для реализации предложенных в п.2.1. Задач.

Выводы по 1 главе:

Программирование – это один из важных разделов курса «Информатика и ИКТ», изучение которого позволяет решать целый ряд дидактических и педагогических задач.

Программирование вырабатывает у учащихся четкое логическое мышление, аккуратность и внимательность, развивает находчивость, изобретательность, фантазию и творческие способности.

Существует несколько сотен используемых языков программирования. Чаще всего в учебных заведениях изучаются Pascal, поскольку изначально задумывался как универсальный язык для начинающих программистов, имеющий основные достоинства, как простота синтаксиса, простота организации данных и управляющих структур.

Программные средства языка Pascal позволяют работать не только с текстовой, но и с графической информацией. Обширный набор процедур и функций Pascal, позволяют создавать разнообразные графические изображения и выводить на экран текстовые надписи. Наглядное представление результатов работы придает осмысленность действиям, возможность выбирать цвета, формы и использовать другие элементы графического режима пробуждает творческую часть личности учащихся, помогает достигать метапредметных и личностных результатов обучения. Все это в совокупности приводит к повышению интереса и познавательной активности.

Глава 2. Дидактические материалы для обучения программированию с использованием средств визуализации

2.1. Система учебных задач по программированию на Паскале

Имеет ли значение графика для обучения начинающих программистов?

Несомненно, все зависит от аудитории. Если программированию обучается школьник, то графика очень важна - мышление конкретное, и хочется сразу видеть результаты своего труда. Абстрактное мышление быстро утомляет. Если программировать учится студент - здесь другая картина. Графика нужна по большей мере как вспомогательное средство для визуализации результатов, динамики выполнения алгоритмов.

Для первичного усвоения нами была разработана система задач, графическая составляющая которых достаточно проста для того, чтобы учащиеся могли целиком сконцентрироваться на понимании новых элементов программирования, но в то же время, графический результат работы позволяет наглядно увидеть плоды своих трудов, что закрепляет усвоение, кроме того графика интереснее сухих цифр, что также положительно влияет на интерес и мотивацию учащихся.

Данная система задач направлена на первичное усвоение основных элементов языка Паскаль.

Рассмотрим предложенную систему задач по разделам:

1. Введение в графику в Паскале

2. Управляющие структуры

3. Подпрограммы

4. Одномерные массивы

5. Работа с файлами (дополнительный)

Раздел 1. Введение в графику в Паскале

Справочный материал по данному разделу представлен в приложении А.

Задания к разделу

Цель: знакомство и приобретение практических навыков по использованию графических примитивов: окружность, прямоугольник и др. использование их свойств.

Для иллюстрации работы с графикой в PascalABC нарисуем снеговика.

Исходный код приложения для решения задачи приведен на листинге 1.2.

Листинг 1.2.

```
Program Example1;
uses GraphABC;
begin
 Brush.Color := clLightBlue;
 Circle(260, 380, 100);
 Circle(260, 220, 60);
 Circle(260, 120, 40);
 Brush.Color := clOrange;
 Rectangle(230, 40, 290, 90);
 Brush.Color := clRed;
 Circle(260, 130, 5);
 Brush.Color := clMaroon;
 Circle(250, 110, 5);
 Circle(270, 110, 5);
 Brush.Color := clWhite;
 TextOut(50, 200, 'SnowMan');
end.
```

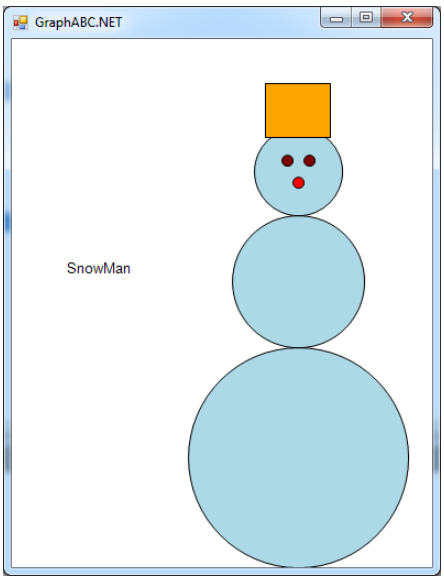

Рисунок 1. Результат решения демонстрационного задания

Самостоятельно выполните следующие задания:

Вариант 1. Нарисуйте персонажа компьютерной игры PacMan и подпишите его.

Вариант 2. Нарисуйте флаг Японии и напишите ее название.

Указания к выполнению работы:

– Создайте шаблон программы согласно листингу 1.
– Задайте цвет графического примитива.
– Нарисуйте графический примитив.
– Задайте цвет следующего элемента рисунка.
– Нарисуйте соответствующий примитив.

Примерный вариант решения задач показан на рисунке 2.

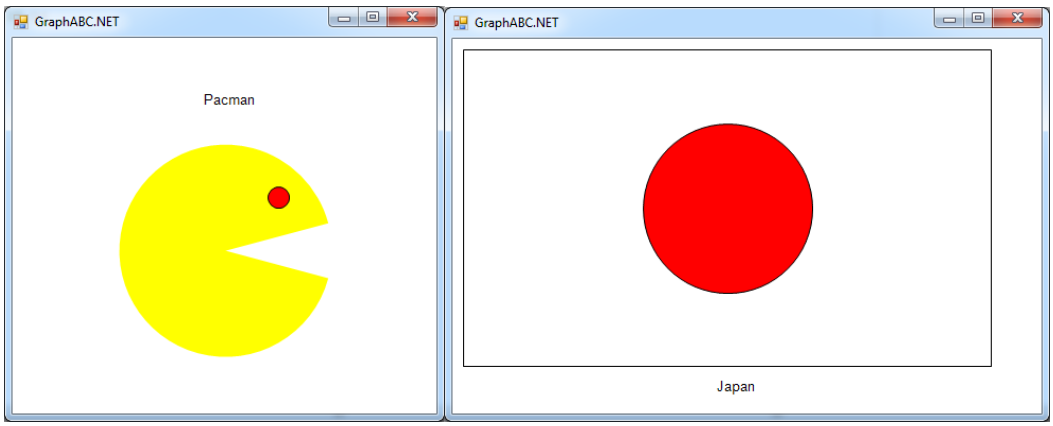

Рисунок 2. Образцы выполнения задания

Раздел 2. Управляющие структуры

Задания к разделу.

После знакомства с теоретическим материалом, справка по которому представлена в приложении Б, учащимся следует предложить следующие задачи:

Задачи на закрепление темы «управляющие структуры» (используются графические примитивы:

1) В зависимости от числа, введенного с клавиатуры, нарисовать соответствующий графический объект (1-окружность, 2-прямоугольник, 3-треугольник).

2) Вывести по горизонтали/вертикали N окружностей одного цвета (разного цвета, с чередованием дух цветов)

Задачи на приобретение навыков работы с условным оператором и циклами в Паскале

Задача 1.

Нарисуем 9 шаров, используя операторы FOR, WHILE и REPEAT … UNTIL. Для задания цвета шаров используем операторы ветвления IF и множественного выбора CASE.

Исходный код приложения для решения задачи приведен на листинге 3.1.

Листинг 3.1.

```
Program Example2;
uses GraphABC;
Var
 I : integer;
begin
 Brush.Color := clLightBlue;
 for I := 1 to 4 do
 begin
   if i > 2 then
    Brush.Color := clBlack;
   Circle(i * 50 , 60, 20);
  end;
 I := 1;
 While I <= 4 do
 begin
   if I > 2 then
    Brush.Color := clLightBlue
   else
     Brush.Color := clBlue;
  Circle(I * 50 , 120, 20);
  I := I + 1;
 end;
 I := 1;
 Repeat
  case I of
    1 : Brush.Color := clLightBlue;
    2 : Brush.Color := clBlue;
    3 : Brush.Color := clGreen;
    4 : Brush.Color := clMagenta;
```

```
    end;
   Circle(I * 50 , 180, 20);
   I := I + 1;
  Until I > 4;

end.
```

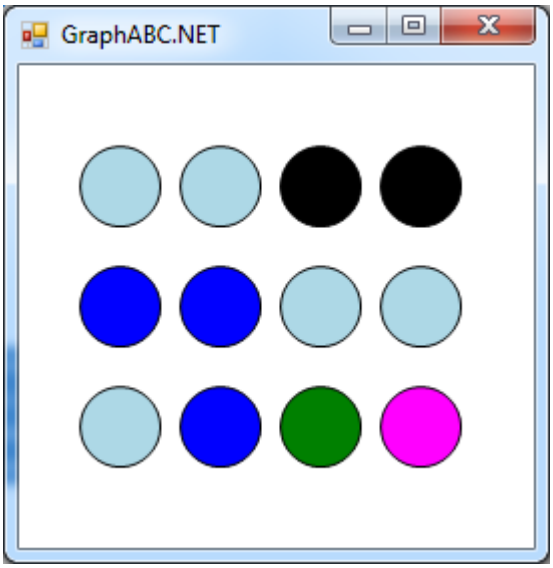

Рисунок 3. Результат решения демонстрационного задания

Самостоятельно выполните следующие задания:

Вариант 1. Нарисуйте три мишени. Для первой мишени используйте цикл FOR, для второй WHILE, для третьей – REPEAT…UNTIL. Используйте операторы IF и CASE по своему усмотрению, однако следует применить каждый хотя бы один раз.

Замечание. Рисование следует начинать с большего круга, т.к. следующие фигуры рисуются поверх предыдущих. Подумайте, как следует для этого использовать переменную цикла.

Вариант 2. Нарисуйте шахматную доску размером 3x3. Для первой линии используйте цикл FOR, для второй WHILE, для третьей –

REPEAT…UNTIL. Используйте операторы IF и CASE по своему усмотрению, однако следует применить каждый хотя бы один раз.

Примерный вариант решения задач показан на рисунке 4.

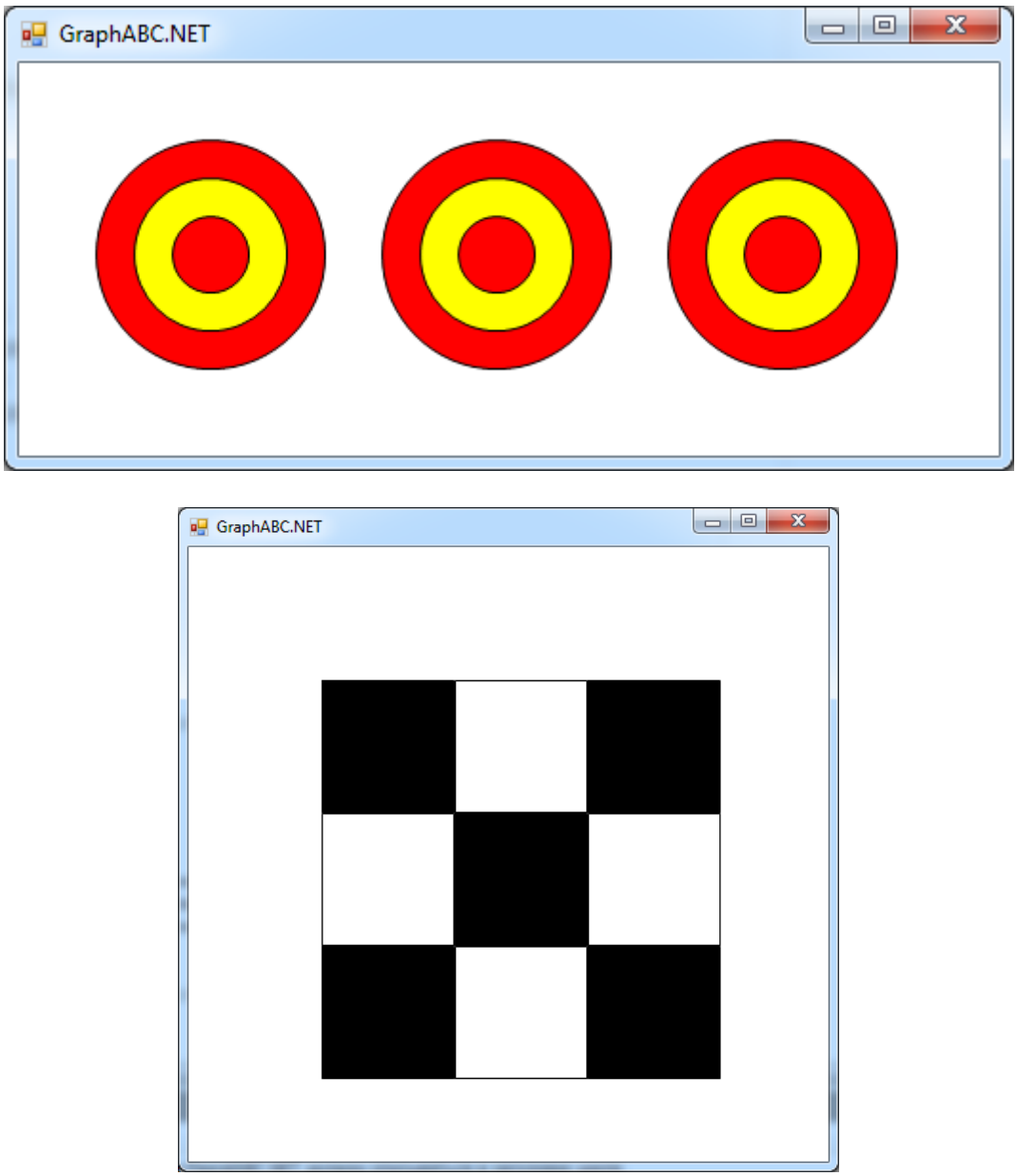

Рисунок 4. Образцы выполнения задания

Раздел 3. Подпрограммы

Справочный материал по данному разделу представлен в приложении В.

Цель: знакомство учащихся с процедурами и функциями .

Задания к разделу.

Для иллюстрации применения процедур и функций перепишем с их применением программу из предыдущего раздела. Функция MyColor будет устанавливать цвет рисунка.

Исходный код приложения для решения задачи приведен на листинге 4.3.

Листинг 4.3.

```
Program Example3;
uses GraphABC;
Var
 I :integer;
 Function MyColor(I : Integer) : Color;
Begin
case I of
    1 : MyColor := clLightBlue;
    2 : MyColor := clBlue;
    3 : MyColor := clGreen;
    4 : MyColor := clMagenta;
   end;
End;
Procedure DrawSomething(X, Y, Z : Integer);
Begin
 Circle(X , Y, Z);
End;
 begin
 Brush.Color := clLightBlue;
 for i := 1 to 4 do
 begin
  Brush.Color := MyColor(i);
    DrawSomething(i * 50 , 60, 20);
  end;
```

```
I := 1;
While I <= 4 do
begin
 Brush.Color := MyColor(i);
 DrawSomething(i * 50 , 120, 20);
 I := I + 1;
end;
I := 1;
Repeat
 Brush.Color := MyColor(i);
 Circle(i * 50 , 180, 20);
 I := I + 1;
Until I > 4;
end.
```

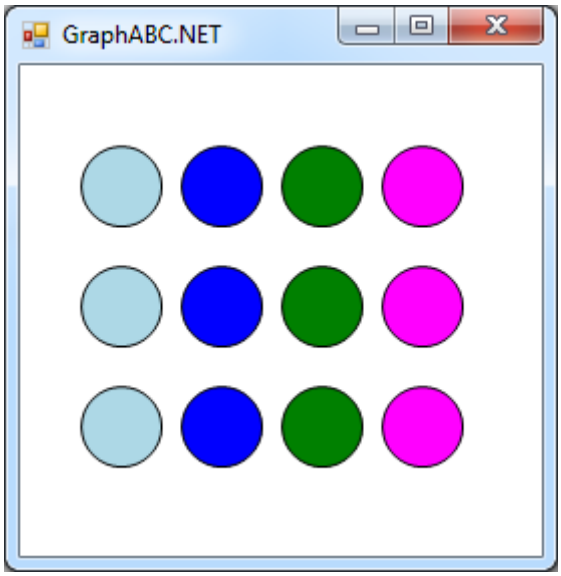

Рисунок 5. Результат решения демонстрационного задания

Самостоятельно модифицируйте свои программы из предыдущего раздела с применением процедур и функций.

Раздел 4. Одномерные массивы

Задания к разделу.

Задачи на закрепление темы «Одномерные массивы» (используются графические примитивы:

1) В массиве хранятся размеры (радиусы) 10 окружностей, нарисовать данные окружности на экране
2) В массиве хранятся оценки по математике учеников 10а класса. Построить столбчатую диаграмму успеваемости

Задачи на приобретение навыков работы с условным оператором и циклами в Паскале

1) В массиве хранятся размеры (радиусы) 10 окружностей, нарисовать данные окружности на экране. Расположить их в порядке возрастания
2) В массиве хранятся оценки по математике учеников 10а класса. Построить столбчатую диаграмму успеваемости, расположив данные в порядке убывания

Раздел 5. Работа с файлами (дополнительный)

Задания к разделу.

Для иллюстрации применения чтения из файла решим следующий пример:

Дан текстовой файл вида:

```
Название1
Количество1
Название 2
Количество2
…
НазваниеN
```

```
КоличествоN
```

Требуется считать данные из файла и вывести их на экран.

Исходный код приложения для решения задачи приведен на листинге 5.1.

Листинг 5.1.

```
Program Example5;
uses GraphABC;
Var
 I, a : integer;
 t : text;
 s : string;

begin
 assign(t,'input.txt');
 reset(t);
 i := 0;
  while eof(t) = false do
   begin
    readln(t, s);
    TextOut(10, i*20, s);
    readln(t, a);
    TextOut(100, i*20, a);
    i := i + 1;
   end;
 close(t);

end.
```

Следует обратить на тип считываемого значения: переменная *a* представляет собой число типа *integer*, переменная *s* – текстовая строка.

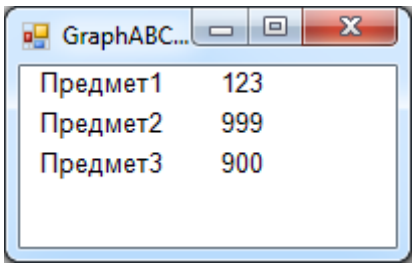

Рисунок 6. Результат решения демонстрационного задания

Самостоятельно выполните следующие задания:

Дан текстовой файл вида:

```
Название1
Количество1
Цена1
Название 2
Количество2
Цена2
…
НазваниеN
КоличествоN
ЦенаN
```

Где *Название* – текстовое значение, *Количество* и *Цена* – целые числа.

Пример файла:

```
Стол
20
30
Стул
40
10
Диван
34
30
```

Вариант 1. Считайте данные из файла. Подсчитайте стоимость для каждого названия и выведите на экран таблицу, содержащую название предмета и его стоимость.

Вариант 2. Считайте данные из файла. Подсчитайте общее количество предметов и выведите список предметов и их общее количество.

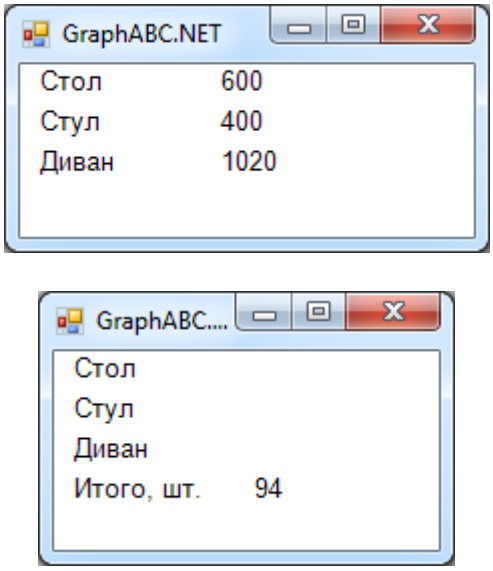

Рисунок 7. Образцы выполнения задания

Итоговое задание

Итоговое задание предназначено для повторения и закрепления всех полученных в данном курсе навыков.

В работе следует использовать как минимум по одной функции из каждого изученного раздела.

Вариант 1. В текстовом файле *in1.txt* храниться 5 записей о названии, количестве, и цене имеющихся на складе магазина конфет.

Выполнить:

- Считать данные из файла;
- Вычислить стоимость конфет для каждого вида конфет.

– По стоимости конфет построить столбчатую диаграмму, каждый столбец нарисовать своим цветом.

– Снизу столбца подписать название конфет.

– Сверху столбца подписать стоимость конфет (позицию для подписи вычислить автоматически).

Вариант 2.

В текстовом файле *in2.txt* хранятся 2 записей об успеваемости учащихся 10-го класса фамилия и две оценки.

Выполнить:

1. Считать данные из файла;
2. Определить максимальную для каждого учащегося.
3. Построить столбчатую диаграмму по максимальной оценке, при этом выделить красным цветом оценки больше 4.
4. Снизу столбца подписать фамилию учащегося.
5. Сверху столбца подписать максимальную балл (позицию для подписи вычислить автоматически).

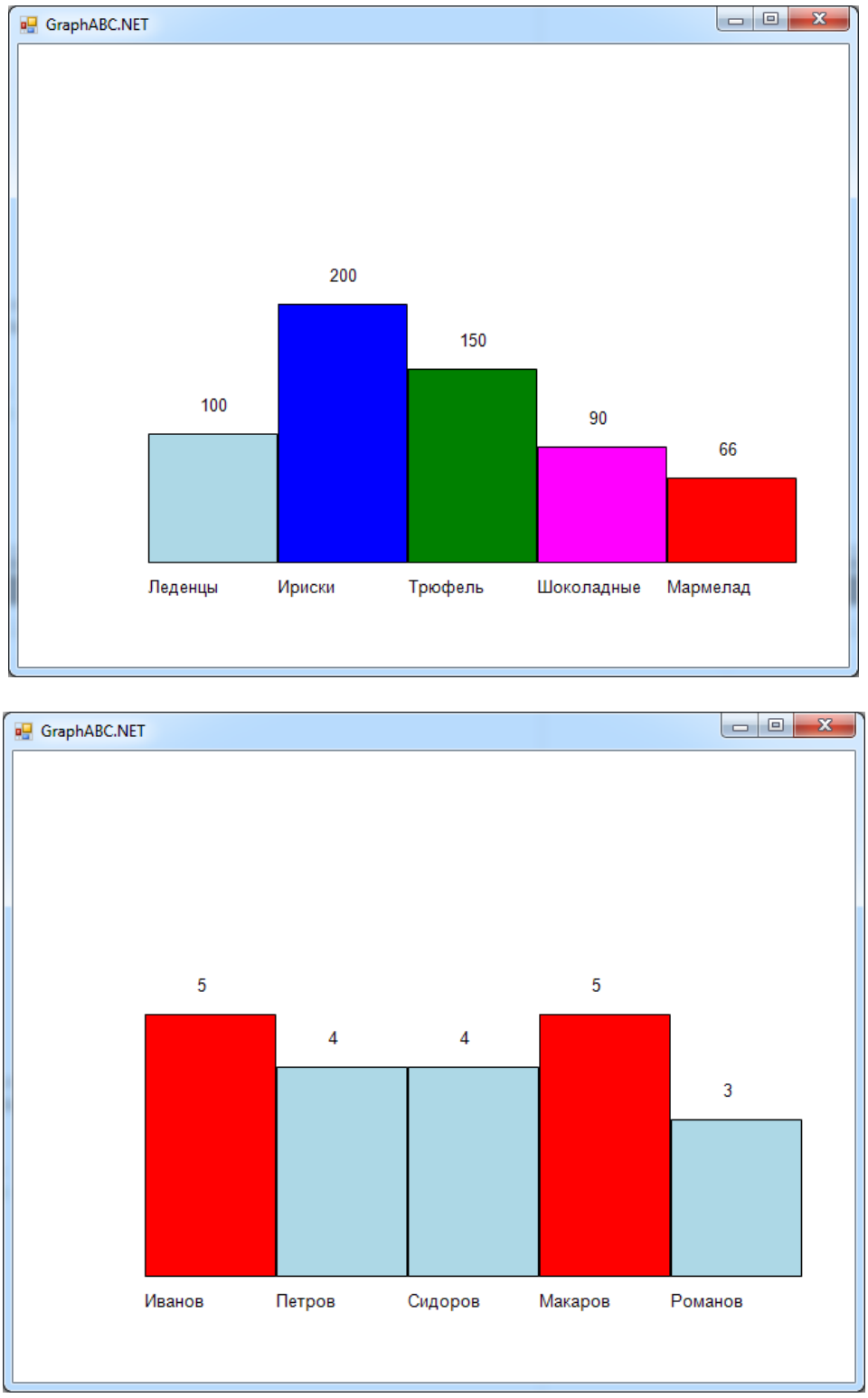

Рисунок 8. Пример возможного решения задания

2.2. Рекомендации по использованию средстввизуализации в обучении школьников программированию

Для успешного изучения данной темы, учитель должен включить следующие разделы:

1. Графические возможности Pascal.

2. Процедуры работы с графическими примитивами.

3. Вывод текста в графическом режиме.

Тема предусматривает выработку значительных практических навыков, поэтому следует обратить внимание на разработку практических заданий.

В начале изучения данной темы можно столкнуться с непониманием учащихся. "А зачем так сложно, если в любом графическом редакторе можно сделать то же самое?" Мотивацией может служить то, что, зная приемы программирования, создаются быстро такие элементы, на которые в графическом редакторе уйдет много времени и не будет возможности многократного повторения и внесения быстрых изменений в рисунок. Например, привести следующую задачу: нарисовать новогоднюю открытку в основе которой будет много елочек разного размера «разбросанных» по экрану [7].

Далее необходимо познакомит учеников с основными понятиями компьютерной графики.

Следующим шагом будет введение графических операторов, которые поддерживает Pascal. При введении операторов, необходимо идти из практических соображений. Ввод оператора должен сопровождаться некой практической задачей, которая после введения оператора будет успешно решаться. Академический подход, при котором перечисляются все операторы и их описание, здесь не допустим.

Следует обратить внимание на связь данной темы с другими разделами темы программирования: циклами, вспомогательными программами,

рекурсией. При решении задач обращать внимание школьников не на запоминание графических операторов и их синтаксиса (это можно найти в любом справочнике), а на алгоритм решения задачи, последовательность действий, оформленность логических кусков программы в процедуры и функции, использование циклов для повторяющихся элементов.

После теоретических моментов школьники выполняют лабораторные работы и сохраняют свои результаты в отдельных папках с последующей их защитой перед преподавателем. При проверке следует также обращать внимание на «красоту» алгоритма, а не рисунка, при необходимости давать ученику на доработку.

При малом количестве часов, одним из методов, позволяющих добиваться положительной мотивации к учению и хороших результатов в изучении данной темы, является метод проектов.

В конце изучения темы проводиться проверочная работа, которая может заключаться в выполнении какого-либо рисунка с последующей проверкой учителем.

Рекомендуется проводить обмен рисунков между школьниками, а также устраивать компьютерный вернисаж и конкурсы. Такие мероприятия стимулируют развитие познавательной и творческой активности.

Выводы по главе 2:

Использование графических возможностей языка программирования в обучении, стимулирует творческую активность школьника, позволяет использовать интересные примеры, это приводит к лучшему усвоению материала. Разработанная система задач основана на использовании графических примитивов в освоении основных разделах курса, что делает результат решения задачи более наглядным и ведет ученика от решения простой задачи к более сложной, например, от построения одной окружности к построению N окружностей, расположенных по горизонтали в порядке возрастания размера.

Заключение

Общеобразовательная сила идей и методов, заимствованных из области программирования, несет в себе огромный потенциал для развития компонентов содержания общего школьного образования.

Процесс изучения программирования связан с мыслительной деятельностью учащихся. При изучении алгоритмов и программирования, у учащихся развивается логическое и рациональное мышления, которые, в свою очередь, оказывают положительное влияние на развития индивида. Программирование позволяет научить учащихся систематизировать свои знания, применять их в решении конкретных задач, оценивать и анализировать получившиеся результаты, учитывая стоящие перед ним задачи и прослеживая возможные последствия принятия решения.

Данная работа была нацелена на разработку дидактических материалов для обучения программированию на Паскале с использованием средств визуализации. Здесь представлен теоретический материал, подлежащий усвоению, разработана система задач по программированию на Паскале с использованием графических возможностей.

Введение в тему «Программирование» изучения графических примитивов, стимулирует творческую активность школьника, позволяет использовать интересные примеры, это приводит к лучшему усвоению материала. Рекомендуется включать данную тему в учебную программу по информатике.

Список литературы

1. Абрамян М.Э., Михалкович С.С. Основы программирования на языке Паскаль: Скалярные типы данных, управляющие операторы, процедуры и функции. — Ростов-на-Дону: ООО «ЦВВР», 2004. — 198 с.

2. Абрамян М.Э., Белякова Ю.В., Михалкович С.С. Использование веб-среды PascalABC.NET для дистанционного обучения программированию // Журнал «Дистанционное и виртуальное обучение». — 2012. — Т. 3. — С. 14—24.

3. В.В. Махно, С.С. Михалкович, М.В. Пучкин. Основы программирования графики. Часть 1. Базовые возможности. Ростов-на-Дону, 2007.

4. Бочкин А.И. Методика преподавания информатики. – Минск: Вышэйшая школа, 2012. – 431 с.

5. Бурцева, Г. А. Графика в обучении программированию / Г.А. Бурцева // Информатика и образование. – 2002. – № 6. – С. 25-42.

6. Гейн А.Г., Юнерман Н.А. Информатика, 10-11: Книга для учителя. – М.: Просвещение, 2001. – 207 с.

7. Дуванов А.А. «Незаметки» Сидорова (Цикл статей) [Текст] // Информатика, №№ 6, 7, 8, 9, 10, 11, 13, 15, 16, 17, 18, 19, 20, 2001 г

8. Епанешников А.М., Епанешников В.А. Программирование в среде TURBO-PASCAL 7.O: учебное пособие. – М.: ДИАЛОГ-МИФИ, 2012. – 282 с.

9. Журбина Н.А. Информационно-коммуникационные технологии в образовании / Н.А. Журбина // Информационное общество. – 2001. – № 2. – С. 5-6.

10. Информатика(Базовый курс) С. В. Симонович, СПб: Питер, 2001г.

11. Каймин В.А. и др. Основы информатики и вычислительной техники: Пробное учебное пособие для 10-11 кл. сред. шк. – М.: Просвещение, 2014. – 272 с.

12. Кудинова, В.И. О пользе программирования для школьников / В.И. Кудинова // Информатика и образование. – 2002. – № 11. – С. 17-28.

13. Лапчик М.П. Методика преподавания информатики: учебник. – М.: Академия, 2003. – 624 с.

14. Марченко А.И. Марченко Л.А. Программирование в среде Turbo Pascal 7.0: учебное пособие. – Киев: Век+, 2009. – 460 с.

15. Методические проблемы обучения программированию в основной и средней школе [электронный ресурс] URL: https://refdb.ru/look/1617587.html (15.06.2017)

16. Назарова Т.С., Полат Е.С. Средства обучения: технология создания и использования. – М.: Изд-во УРАО, 2004. – 204 с.

17. Основы языка Turbo Pascal(учебный курс), П.И. Рудаков, М.А. Федотов, Москва: Радио и Связь, 2013г.

18. Оценка качества подготовки выпускников средней (полной) школы по информатике / А.А. Кузнецов, Л.Е. Самовольнова, Н.Д. Угринович. – М.: Дрофа, 2001. – 64 с.

19. Основы программирования: С. Окулов -- Москва, Бином. Лаборатория знаний, 2010 г.

20. Попов В.Б. Turbo Pascal для школьников: учебное пособие / В.Б. Попов. – М.: Финансы и статистика, 2012. – 234 с.

21. Практика программирования, Ю. Кетков, А. Кетков, СПб: БХБ/ Петербург, 2002г.

22. Программирование. С. Симонович, Г. Евсеев, Москва: АСТ - ПРЕСС книга 2000г.

23. Семакин И. Г., Шестаков А. П. Основы программирования: учебное пособие. – М.: Лаборатория Базовых Знаний, 2003. – 312 с.

24. Софронова Н.В. Теория и методика обучения информатике. – М.: Высшая школа, 2004. – 223 с.

25. Угринович Н.Д. Информатика и информационные технологии: учебник для 10-11классов. – М.: БИНОМ. Лаборатория знаний, 2005. – 511 с.

26. Угринович Н.Д. Преподавание курса «Информатика и ИКТ» в основной и старшей школе: методическое пособие. – М.: БИНОМ. Лаборатория знаний, 2004. – 139 с.

27. Хуторской А.В. Практикум по дидактике и современным методикам обучения. – СПб.: Питер, 2004. – 541 с.

28. Цветкова М.С. Информатика в начальной, основной и профильной школе // Информатика и образование. – 2002. – № 1. – С. 9.

29. Челак Е.Н., Конопатова Н.К. Развивающая информатика. Методическое пособие. – М.: Лаборатория Базовых Знаний, 2001. – 208 с.

30. Turbo Pascal: учебное пособие / под ред. С.А. Немнюгин, – Спб.: Питер, – 2000. – 496 с.

Приложение А

Справочный материал по разделу «Введение в графику в Паскале»

Для рисования различных геометрических фигур в Паскале нам понадобится среда PascalABC.Net или PascalABC.

При работе с графическим окном нужно учитывать две особенности.

- начало координат – точка (0, 0) – находится не где-то посредине окна, а в левом верхнем углу.
- положительное направление оси OY показывает не вверх, а вниз (OX направлена вправо).

Графический экран PascalABC (по умолчанию) содержит 640 точек по горизонтали и 400 точек по вертикали.

Для работы в графическом режиме необходимо подключение модуля GraphABC.

Структура минимальной программы на языке PascalABC приведена на листинге 1.1.

Листинг 1.1.

```
program Unit1;
uses GraphABC;
begin


end.
```

При запуске такой программы возникает специальное графическое окно, и все рисование происходит именно на нем. Рисование производиться с

помощью графических примитивов, которые представляют собой процедуры, осуществляющие рисование в графическом окне. Рисование осуществляется текущим пером (линии), текущей кистью (заливка замкнутых областей) и текущим шрифтом (вывод строк).

Приведем основные процедуры для работы с графикой в PascalABC для решения задач этого раздела:

- *procedure TextOut(x,y: integer; s: string);* Выводит строку s в прямоугольник к координатами левого верхнего угла (x,y)
- *procedure TextOut(x,y: integer; n: integer);* Выводит целое n в прямоугольник к координатами левого верхнего угла (x,y)
- *procedure FillPie(x,y,r,a1,a2: integer);* Заполняет внутренность сектора окружности, ограниченного дугой с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки)
- *procedure DrawPie(x,y,r,a1,a2: integer);* Рисует сектор окружности, ограниченный дугой с центром в точке (x,y) и радиусом r, заключенной между двумя лучами, образующими углы a1 и a2 с осью OX (a1 и a2 – вещественные, задаются в градусах и отсчитываются против часовой стрелки)
- *procedure FillCircle(x,y,r: integer);* Заполняет внутренность окружности с центром (x,y) и радиусом r
- *procedure DrawCircle(x,y,r: integer);* Рисует окружность с центром (x,y) и радиусом r
- *procedure FillRectangle(x1,y1,x2,y2: integer);* Заполняет внутренность прямоугольника, заданного координатами противоположных вершин (x1,y1) и (x2,y2)

– *procedure Rectangle(x1,y1,x2,y2: integer);* Рисует заполненный прямоугольник, заданный координатами противоположных вершин (x1,y1) и (x2,y2).

Более подробную информацию о графических примитивах можно получить к справке PascalABC.

Также нам понадобиться применение функций для установки цвета кисти, «которой» мы рисуем графические примитивы Brush.Color := ЦВЕТ, где ЦВЕТ задается из ряда цветовых констант, например clRed, clWhite, clGreen, clYellow и прочих (полный список констант содержится в справке PascalABC. Примечание: для быстрого вызова раздела справки можно написать известную команду или константу из этого раздела и установив на нее курсов, нажать кнопку F1 – сразу откроется соответствующий раздел справки).

Приложение Б

Справочный материал по разделу «Составные операторы»

Оператор в программе – это единое и неделимое предложение, выполняющее какое-либо действие. Типичный простой оператор – это оператор присваивания. Другим примером может служить вызов какой-либо процедуры в программе. Важно, что под любым оператором подразумевается действие (присваивание, вызов подпрограммы и т.п.). Блоки описания переменных, констант, типов и меток не являются в этом смысле операторами.

Два последовательных оператора обязательно должны разделяться точкой с запятой «;».

Листинг 2.1. Примеры простых операторов

```
a := 10;
b := a*5;
Write( a, b );
```

Если какое-то действие мыслится как единое, но реализуется несколькими различными операторами, то последние могут быть представлены как составной оператор.

Определение. Составной оператор - объединение нескольких операторов в одну группу. Группа операторов внутри составного оператора заключается в операторные скобки (begin-end).

Листинг 2.2.

```
begin
 a := 10;
 b :+ a*5;
 Write( a, b );
end;
```

Составной оператор может содержать любое количество простых операторов. Он допускает вложенность, т.е. может содержать внутри себя другие составные операторы.

ВАЖНО: Составной оператор применяется в тех случаях, когда синтаксис языка Паскаль допускает использование только одного оператора, в то время как алгоритм требует задания некоторой последовательности действий. В Паскале все управляющие структуры (операторы) не различают простой и составной оператор: там где стоит простой оператор, можно поставить и составной.

Приложение В

Справочный материал по разделу «Управляющие структуры»

Знание синтаксиса языка лишь теоретически позволяет приступить к созданию программ. Дело в том, что важнейшей частью любого языка программирования, во многом определяющими удобство составления алгоритмов, являются его управляющие структуры - операторы, или инструкции.

В реальных программах выполнение операций не бывает строго последовательным: постоянно требуются различные переходы, ветвления, повторения и т.д.

Точки алгоритма, в которых выполняется выбор дальнейшего хода программы, называются точками выбора. Выбор очередного шага решения задачи осуществляется в зависимости от выполнения некоторого условия.

Собственно за переход, в классическом варианте, отвечает небезызвестная инструкция безусловного перехода goto, которая в Pascal используется совместно с метками, декларируемыми в заголовочной части программы при помощи ключевого слова label. Использование инструкции безусловного перехода восходит корнями к тем временам, когда создавались первые высокоуровневые языки программирования, в том числе и Pascal. Она досталась им в наследство от низкоуровневых языков типа Assembler, в которых описание программы создавалось в виде, удобным для машины. Но на сегодня такой подход уже не востребован и вышел из употребления, вместе с безусловным переходом и инструкцией goto. Оператор GOTO противоречит принципам технологии структурного программирования. Современные языки программирования не имеют в своем составе такого оператора, и в его использовании нет необходимости. Кроме того в современных компьютерах используется так называемый конвейерный способ. Если в программе встречается оператор безусловного перехода, то

такой оператор ломает весь конвейер, заставляя создавать его заново, что существенно замедляет вычислительный процесс.

К управляющим структурам в Pascal относятся:

- условные операторы;
- оператор выбора;
- оператор повторения с параметром;
- оператор повторения с предусловием;
- оператор повторения с постусловием.

Условный оператор используется в программе для реализации алгоритмической структуры – ветвления. В данной структуре вычислительный процесс может продолжаться по одному из возможных направлений. Выбор направления обычно осуществляется проверкой какого-либо условия. Существует два вида структуры ветвления (рисунок 3): структура вилка и обход.

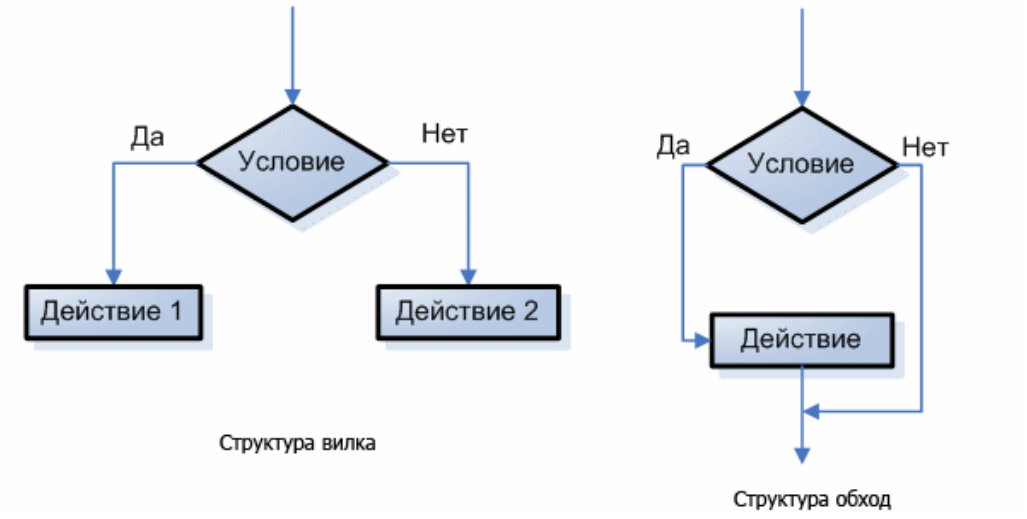

Рисунок 3. Структуры ветвления

В языке Паскаль условный оператор IF это средство организации ветвящегося вычислительного процесса.

Формат:

IF [логическое_выражение] Then [оператор_1]; Else [оператор_2];

IF, Then, Else – служебные слова. [оператор_1], [оператор_2] – обыкновенные операции языка Паскаль. Часть Else является необязательной (может отсутствовать).

Оператор IF работает следующим образом: вначале проверяется результат логического выражения. Если результат Истина(TRUE), то выполняется [оператор_1], следующий за служебным словом Then, а [оператор_2] пропускается. Если результат Ложь(FALSE), то [оператор_1] пропускается, а [оператор_2] исполняется.

Если часть Else отсутствует, то оператор IF имеет не полную форму:

IF [логическое_выражение] Then [оператор];

В этом случае, если результат Истина(TRUE), то выполняется [оператор], если Ложь(FALSE), то управление передается оператору, следующему за оператором IF.

Оператор выбора CASE. Предназначен для реализации множественных ветвлений, поскольку оператор IF может реализовать всего два направления вычислительного процесса, использовать его для реализации множественных ветвлений не всегда удобно. Множественное ветвление реализуется оператором CASE.

Формат:

CASE [ключ_выбора] OF

[константа_выбора_1]:[оператор_1];

[константа_выбора_2]:[оператор_2];

...

[константа_выбора_N]:[оператор_N];

ELSE [оператор];

End;

CASE, OF, ELSE, END – служебные слова. [ключ_выбора] – это параметр одного из порядковых типов. [константы_выбора] – константы того же типа, что и ключ выбора, реализующие выбор. [оператор_1(N)] – обыкновенный оператор. ELSE может отсутствовать.

Оператор выбора работает следующим образом: до работы оператора определяется значение параметра ключ выбора. Этот параметр может быть либо выражен как переменная в программе, либо другим путем. Затем параметр ключ выбора последовательно сравниваем с константой выбора. При совпадении значения ключа выбора с одной из констант выбора, выполняется оператор, следующий за этой константой, а все прочие операторы игнорируются. В случае не совпадения ключа выбора ни с одним из констант, выполняется оператор, следующий за Else. Часто Else является не обязательной и в случае несовпадения ключа выбора ни с одной из констант выбора и при отсутствии Else, управление передается оператору, следующему за оператором CASE.

В операторе CASE нет явной проверки условия, характерного для оператора IF. В тоже время в неявном виде операция сравнения выполняется.

Оператор повторения с параметром. В цикле с параметром всегда имеются так называемые параметры цикла: X, X_n, X_k, ΔX. Иногда цикл с параметром называют регулярным циклом. Характерной чертой является то, что число циклов и повторений можно определить до выполнения цикла.

Формат:

FOR [параметр_цикла] := [н_з_п_ц] To [к_з_п_ц] Do [оператор];

FOR, To, Do – служебные слова. [параметр_цикла] – параметр цикла. [н_з_п_ц] – начальное значение параметра цикла. [к_з_п_ц] – конечное значение параметра цикла. [оператор] – произвольный оператор.

Параметр цикла должен быть переменой порядкового типа. Начальное и конечное значения параметра цикла должны быть того же типа, что и параметр цикла.

Оператор повторения с предусловием относиться к итерационным циклам. В итерационном цикле невозможно определить число циклов до его выполнения. Он выполняется до тех пор, пока выполняется условие продолжение цикла.

Формат: *WHILE [условие] Do [оператор];*

WHILE, Do – служебные слова. [условие] – выражение логического типа. [оператор] – обыкновенный оператор.

Оператор While работает следующим образом: вначале работы проверяется результат логического условия. Если результат истина, то выполняется оператор, после которого осуществляется возврат на проверку условия с новым значением параметров в логическом выражении условия. Если результат ложь, то осуществляется завершение цикла.

При работе с While надо обратить внимание на его свойства:

1. условия, использованные в While, являются условием продолжения цикла;
2. в теле цикла всегда происходит изменение значения параметра входящего в выражение условия;
3. цикл While может, не выполнится ни разу, поскольку проверка условия в продолжение цикла выполняется до тела цикла.

Оператор повторения с постусловием также относиться к итерационным циклам.

Формат: *REPEAT [тело_цикла]; UNTIL [условие];*

Оператор REPEAT работает следующим образом: сначала выполняются операторы тела цикла, после чего результат проверяется логического условия. Если результат ложь, то осуществляется возврат к выполнению операторов очередного тела цикла. Если результат истина, то оператор завершает работу.

Оператор Repeat имеет следующие особенности:

- в Repeat проверяется условие завершения цикла и если условие выполняется, то цикл прекращает работу;
- тело цикла всегда выполняется хотя бы один раз;
- параметр для проверки условия изменяется в теле цикла;

Замечания.

В строке №3 мы записываем служебное слово Var. Оно используется для объявления переменных.

Переменные – это различные значения, числа или слова, которые могут меняться в процессе выполнения программы. Когда мы вводим с клавиатуры числа или буквы, они записываются в переменные.

После слова Var через пробел указываем идентификатор переменной (т.е ее название, которое мы придумываем сами). Переменные – это не служебные слова, программист задает их сам. В данном случае мы задали одну переменную «I» и в дальнейшем мы будем использовать только эту переменную. После записи переменной через двоеточие указывается тип данных. Существует несколько типов данных Один из них - Integer (целый). Он дает понять программе, что наша переменная «n» может быть только целым числом, лежащим в диапазоне от -2147483648 до 2147483647до.

Использование различных типов данных зависит от конкретных потребностей программиста при написании программы. Самое главное на данном этапе понять, что если в своей программе вы собираетесь использовать какое-то число, то для него нужно указать переменную (в нашем случае «I») и тип данных (в нашем случае Integer).

Также следует обратить внимание на применение рассмотренного в предыдущем разделе составного оператора – операторы заключены в операторные скобки begin-end;

Приложение Г

Справочный материал по разделу «Подпрограммы»

В языке Паскаль, как и в большинстве языков программирования, предусмотрены средства, позволяющие оформлять вспомогательный алгоритм как подпрограмму. Это бывает необходимо тогда, когда какой-либо подалгоритм неоднократно повторяется в программе или имеется возможность использовать некоторые фрагменты уже разработанных ранее алгоритмов. Кроме того, подпрограммы применяются для разбиения крупных программ на отдельные смысловые части в соответствии с модульным принципом в программировании.

В языке Паскаль имеется два вида подпрограмм - процедуры и функции.

Процедура или функция представляет собой последовательность операторов, которая имеет имя, список параметров и может быть вызвана из различных частей программы. Функции, в отличие от процедур, в результате своего выполнения возвращают значение, которое может быть использовано в выражении. Для единообразия функции и процедуры называются подпрограммами.

Любая используемая в программе процедура или функция должна быть предварительно описана в разделе описаний.

Описание процедуры имеет вид:

procedure имя(список формальных параметров);

раздел описаний

begin

операторы

end;

Описание функции имеет вид:

function имя(список формальных параметров): тип возвращаемого значения;

раздел описаний

begin

операторы

end;

Операторы подпрограммы, окаймленные операторными скобками begin/end, называются телом этой подпрограммы.

Список фактических параметров - это их перечисление через запятую. При вызове фактические параметры как бы подставляются вместо формальных, стоящих на тех же местах в заголовке. Таким образом происходит передача входных параметров, затем выполняются операторы исполняемой части процедуры, после чего происходит возврат в вызывающий блок. Передача выходных параметров происходит непосредственно во время работы исполняемой части.

Вызов функции может производиться аналогичным способом, кроме того имеется возможность осуществить вызов внутри какого-либо выражения. В частности имя функции может стоять в правой части оператора присваивания, в разделе условий оператора if и т.д.

Для передачи в вызывающий блок выходного значения функции в исполняемой части функции перед возвратом в вызывающий блок необходимо поместить следующую команду:

имя функции := результат;

При вызове процедур и функций необходимо соблюдать следующие правила:

1. количество фактических параметров должно совпадать с количеством формальных;
2. соответствующие фактические и формальные параметры должны совпадать по порядку следования и по типу.

Листинг 4.1. Пример описания и вызова процедуры:

```
Program Example_of_Procedure;
uses GraphABC;

Procedure DrawSomething(X, Y, Z: Integer);
Begin
 Circle(X , Y, Z);
End;

Begin
 DrawSomething( 100, 200, 20);
End.
```

Листинг 4.2. Пример описания и вызова функции

```
Program Example_of_Function;
uses GraphABC;

Function MyColor(I : Integer) : Color;
Begin
  case i of
    1 : MyColor := clRed;
    2 : MyColor := clYellow;
    3 : MyColor := clRed;
    4 : MyColor := clYellow;
```

```
    end;
End;
Begin
Brush.Color := MyColor(1);
  End.
```

Приложение Д

Справочный материал по разделу «Работа с файлами»

Файл - это упорядоченная последовательность однотипных компонентов, расположенных на внешнем носителе. Файлы предназначены только для хранения информации, а обработка этой информации осуществляется программами. Использование файлов целесообразно в случае:

3. долговременного хранения данных ;
4. доступа различных программ к одним и тем же данным;
5. обработки больших массивов данных, которые невозможно целиком разместить в оперативной памяти компьютера

В Паскале определены текстовые файлы, типизированные и нетипизированные. Файл, не содержащий ни одного элемента, называется пустым. Создается файл путем добавления новых записей в конец первоначально пустого файла. Длина файла, т.е. количество элементов, не задается при определении файла.

Все файлы должны быть описаны в программе либо в разделе переменных VAR, либо в разделе типов TYPE. Под чтением файла понимают ввод данных из внешнего файла, находящегося на диске, в оперативную память машины. Запись в файл - вывод результатов работы программы из оперативной памяти на диск в файл.

Операции для работы с файлами, необходимые в рамках данного раздела, выполняются следующими процедурами:

procedure Assign(f: файл; name: string); Связывает файловую переменную с файлом на диске.

procedure Reset(f: Text); Открывает текстовый файл на чтение в кодировке Windows.

procedure Close(f: файл); Закрывает файл.

function Eof(f: файл): boolean; Возвращает True, если достигнут конец файла.

Для того, чтобы организовать ввод данных из файла нам надо:

Завести переменную типа текст и переменную в которую будем записывать прочитанное значение.

```
Var
  t:text;
  a:integer;
```

Связать эту текстовую переменную с файлом из которого будет производиться чтение.

```
Begin
  assign(t,'input.txt');
```

Открыть файл для чтения.

```
reset(t);
```

Теперь можно читать данные.

```
read(t,a);
```

Или

```
readln(t,a);
```

Процедуры *Read* и *ReadLn* выполняют чтение информации из устройства стандартного ввода. В консольных приложениях этим устройством может быть, например, клавиатура (точнее введённые с клавиатуры данные), в графических приложениях (таких как наше) файл на диске.

Процедуры *Read* и *ReadLn* выполняют схожие действия. Основное отличие между ними следующее: процедура *ReadLn* после завершения ввода выполняет перевод строки (а в случае с файлами читает файл строка за строкой). А процедура *Read* читает данные подряд - без перевода строки.

Как только мы прочитали все переменные необходимо закрыть файл.

```
close(t);
```