СОЗДАНИЕ ПРОДУКТА В DELPHI 7

Ульянов М.В.

г. Пермь, МАОУ «СОШ №30», 11 «а» класс

Руководитель: Лабукина Н.А., г. Пермь, МАОУ «СОШ №30», учитель математики и информатики

Данная статья является реферативным изложением основной работы. Полный текст научной работы, приложения, иллюстрации и иные дополнительные материалы доступны на сайте VI Международного конкурса научно-исследовательских и творческих работ учащихся «Старт в науке» по ссылке: https://school-science.ru/6/4/36831.

Сейчас программы используются почти везде, от простых калькуляторов до тяжёлого машиностроения. Программы внедрены в наш быт. Теперь мы не можем представить себе жизнь без них. На данный момент профессия программиста очень востребована, потому что существует потребность в прогрессе, для которого необходимо большое количество программ, каждая из которых лучше прежней.

Но это всё в будущем. Сегодня, как ученик десятого класса, я могу создать такую программу для своих одноклассников, с помощью которой они могли бы закрепить пройденный материал. Ведь повторение пройденного с использованием программ – проще, удобнее и интереснее, чем решение примеров из учебника.

Программы выполняют всего одну функцию – сокращают время, затрачиваемое на какую-либо операцию. В современном мире не существует задач, которые не способен решить человек. Однако, мы тратим на это слишком много времени. Программы экономят время, которое мы проводим по своему усмотрению. Я хочу сэкономить кому-нибудь пару минут, чтобы можно было провести их со своей семьёй.

Для работы будут использоваться как печатные ресурсы, так и интернет-ресурсы. Мною будет изучен язык программирования Delphi и с его помощью будет создан готовый продукт.

На написание этой работы меня подтолкнули следующие проблемы:

Неумение работать в программе Delphi 7.

Область применения языка программирования Pascal, изучаемого в курсе десятого класса.

Цель работы:

1. Освоение программирования в Delphi 7.
2. Создание рабочего продукта.

Задачи работы:

1. Приобрести Delphi 7.
2. Изучить способ программирования в Delphi 7.
3. Разработать тест.
4. Обучиться азам программирования в Delphi 7.
5. Разобраться, что такое программирование.
6. Разобрать что такое языки программирования.
7. Изучить виды языков программирования.
8. Рассмотреть историю программирования.
9. Ответить на вопрос «Зачем в курсе десятого класса проходят программирование в Pascal?».
10. Научиться видеть ошибки в коде программы.
11. Изучение возможного применения Delphi 7.

1. Теоретическое обоснование темы исследования

1.1. Программирование. Языки программирования

Программирование – процесс создания компьютерных программ, также программирование называют кодированием.

В узком смысле под программированием понимают написание инструкций (программ) на конкретном языке программирования (часто уже по имеющемуся алгоритму или методу решения поставленной задачи). Соответственно, люди, которые этим занимаются, называются программистами, а те, кто разрабатывают алгоритмы – алгоритмистами, специалистами конкретной предметной области.

В более широком смысле под программированием понимают весь спектр деятельности, связанный с созданием и поддерживанием в рабочем состоянии программ – программного обеспечения ЭВМ. Иначе это называется «программная инженерия» («инженерия ПО»). Сюда входят анализ и постановка задачи, проектирование программы, построение алгоритмов, разработка структур данных, написание текстов программ, отладка и тестирование

программы (испытания программы), документирование, настройка (конфигурирование), доработка и сопровождение.

Программирование для ЭВМ основывается на использовании языков программирования, на которых записывается программа. Чтобы она могла быть понята и исполнена ЭВМ, требуется специальный инструмент – транслятор.

В настоящее время активно используются интегрированные среды разработки, включающие в свой состав редактор для ввода и редактирование текстов программ, отладчики для поиска и устранения ошибок, трансляторы с различных языков программирования, компоновщики для сборки программы из нескольких модулей и другие служебные модули.

История программирования

Антикитерский механизм (рис. 1) из Древней Греции был калькулятором, использовавшим шестерни различных размеров и конфигурации, обусловливавших его работу, по отслеживанию метонова цикла (рис. 2), до сих пор использующегося в лунно-солнечных календарях. Аль-Джазари построил программируемый автомат-гуманоид (рис. 3) в 1206 году. Одна система, задействованная в этих устройствах, использовала зажимы и кулачки, помещённые в деревянный ящик в определённых местах, которые последовательно задействовали рычаги, которые, в свою очередь, управляли ударными инструментами.

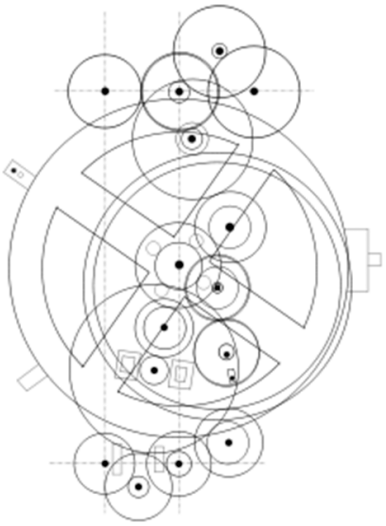

Рис. 1. Антикитерский механизм

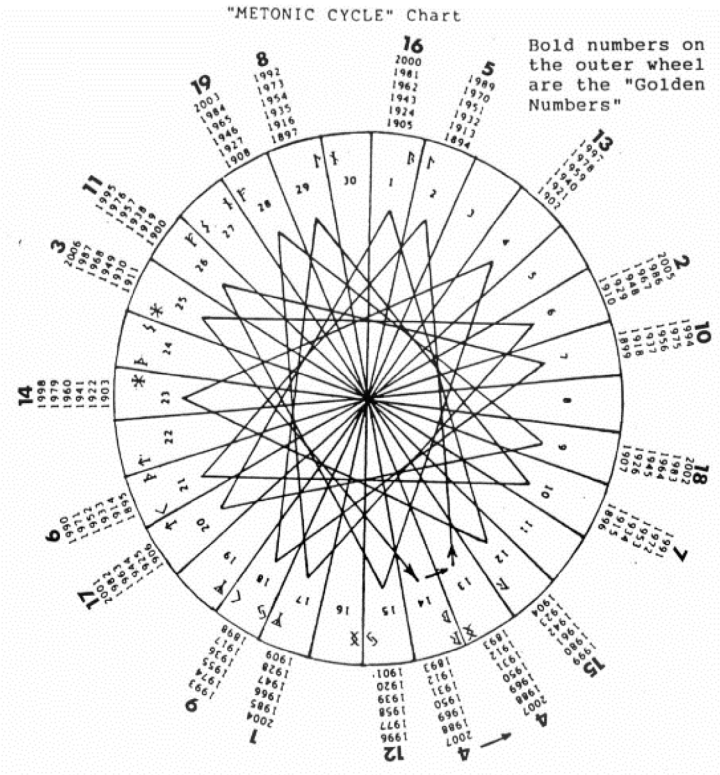

Рис. 2. Метонов цикл

Рис. 3. Программируемый автомат-гуманоид Аль-Джазари

Первым программируемым устройством принято считать жаккардовый ткацкий станок (См. Рис.4), построенный в 1804 году Жозефом Мари Жаккаром, который произвёл революцию в ткацкой промышленности, предоставив возможность программировать узоры на тканях при помощи перфокарт.

Рис. 4. Жаккардовый ткацкий станок

Первое программируемое вычислительное устройство, Аналитическую машину, разработал Чарлз Бэббидж (но не смог её построить). 19 июля 1843 года графиня Ада Августа Лавлейс, дочь великого английского поэта Джорджа Байрона, как принято считать, написала первую в истории человечества программу для Аналитической машины. Эта программа решала уравнение Бернулли, выражающее закон сохранения энергии движущейся жидкости. В своей первой и единственной научной работе Ада Лавлейс рассмотрела большое число вопросов. Ряд высказанных ею общих положений (принцип экономии рабочих ячеек памяти, связь рекуррентных формул с циклическими процессами вычислений) сохранили свое принципиальное значение и для современного программирования. В материалах Бэббиджа и комментариях Лавлейс намечены такие понятия, как подпрограмма и библиотека подпрограмм, модификация команд и индексный регистр, которые стали употребляться только в 1950-х годах. Однако ни одна из программ написанных Адой Лавлейс никогда так и не была запущена.

В 1940-е годы, появились электрические цифровые компьютеры и был разработан язык, который можно считать первым высокоуровневым языком программирования для ЭВМ – «Планкалкюль», созданный немецким инженером К. Цузе в период с 1943 по 1945 годы.

Совершенствование программирования

В период 1960-х – 1970-х годов были разработаны основные парадигмы языков программирования, используемые в настоящее время, хотя во многих аспектах этот процесс представлял собой лишь улучшение идей и концепций, заложенных еще в первых языках третьего поколения.

Язык APL оказал влияние на функциональное программирование и стал первым языком, поддерживавшим обработку массивов.

Язык NPL был разработан в 1960-х годах как объединение лучших черт Фортрана и Кобола.

Язык Симула, появившийся примерно в это же время, впервые включал поддержку объектно-ориентированного программирования. В середине 1970-х группа специалистов представила язык Smalltalk, который был уже всецело объектно-ориентированным.

В период с 1969 по 1973 годы велась разработка языка Си, популярного и по сей день и ставшего основой для множества последующих языков, например, столь популярных, как C++ и Java.

В 1972 году был создан Пролог – наиболее известный (хотя и не первый, и далеко не единственный) язык логического программирования.

В 1973 году в языке ML была реализована расширенная система полиморфной типизации, положившая начало языкам функционального программирования.

Каждый из описанных языков породил по семейству потомков, и большинство современных языков программирования в конечном счете основано на одном из них.

Существует 2 способа реализации языков программирования: компилируемые или интерпретируемые.

Программа на компилируемом языке при помощи компилятора (особой программы) преобразуется в машинный код (набор инструкций) для данного типа процессора и далее собирается в исполнимый модуль, который может быть запущен на исполнение как отдельная программа. Другими словами, компилятор переводит исходный текст программы с языка программирования высокого уровня в двоичные коды инструкций процессора.

Если программа написана на интерпретируемом языке, то интерпретатор непосредственно выполняет исходный текст без предварительного перевода. При этом программа остаётся на исходном языке и не может быть запущена без интерпретатора. Процессор компьютера, в этой связи, можно назвать интерпретатором для машинного кода.

Но это разделение является условным. Так для любого компилируемого языка можно написать интерпретатор. А для любого интерпретируемого языка можно создать компилятор.

Так же интерпретаторы и компиляторы можно отнести к языкам программирования низкого уровня.

Компиляторы – превращают текст программы в машинный код, который можно сохранить и после этого использовать уже без компилятора.

Интерпретаторы – превращают часть программы в машинный код, выполняют его, после этого переходят к следующей части. При этом каждый раз при выполнении программы используется интерпретатор.

Эти языки используются для написания небольших системных программ, драйверов устройств, модулей стыков с нестандартным оборудованием, программирование специализированных микропроцессов, когда важнейшим требованием является компактность, быстродействие и возможность прямого доступа к аппаратным ресурсам.

Ассемблер – язык низкого уровня, широко применяемый до сих пор.

Так же существуют языки программирования высокого уровня. Особенности конкретных архитектур в них не учитываются, поэтому созданные приложения легко переносятся с компьютера на компьютер. В большинстве случаев достаточно просто прокомпилировать программу под определённую компьютерную архитектурную и операционную систему. Разрабатывать программы на таких языках гораздо проще и ошибок можно допускать в разы меньше. Значительно сокращается время разработки программы, что очень важно при разработке больших программных проектов.

Примерами языков программирования высокого уровня являются:

- Delphi
- Pascal
- Java
- C
- Basic
- C++
- Objective-C
- Smalltalk
- C#

Недостатком некоторых языков программирования высокого уровня является большой размер программ в сравнении с программами на языках низкого уровня. Поэтому, в основном, языки программирования высокого уровня применяются для ЭВМ с большим объемом памяти.

1.2. Подробнее о Delphi

Delphi – императивный, структурированный, объектно-ориентированный язык программирования, диалект Object Pascal.

Разберём по частям.

Императивное программирование – это парадигма, которая описывает процесс вычисления в виде инструкций, изменяющих состояние программы. Императивная программа очень похожа на приказы, выражаемые повелительным наклонением в естественных языках, то есть это последовательность команд, которые должен выполнить компьютер.

Структурное программирование – методология разработки программного обеспечения, в основе которой лежит представление программы в виде иерархической структуры блоков. Предложена в 1970-х годах Эдсгером Дейкстром.

В соответствии с данной методологией любая программа строится без использования оператора goto из трёх базовых управляющих структур: последовательность, ветвление, цикл; кроме того, используются подпрограммы. При этом разработка программы ведётся пошагово, методом «сверху вниз».

Методология структурного программирования появилась как следствие возрастания сложности решаемых на компьютерах задач, и соответственно, усложнения программного обеспечения. В 1970–е годы объёмы и сложность программ достигли такого уровня, что традиционная (неструктурированная) разработка программ перестала удовлетворять потребностям практики. Программы становились слишком сложными, чтобы их можно было нормально сопровождать. Поэтому потребовалась систематизация процесса разработки и структуры программ.

Методология структурной разработки программного обеспечения была признана «самой сильной формализацией 70-х годов».

Объектно-ориентированное программирование (ООП) – парадигма, в которой основами концепции являются понятия объектов и классов. В случае языков с прототипированием вместо классов используются объекты-прототипы.

В центре ООП находится понятие объекта. Объект – это сущность, которой можно посылать сообщения и которая может на них реагировать, используя свои данные. Объект – это экземпляр класса. Данные объекта скрыты от остальной программы. Сокрытие данных называется инкапсуляцией.

Наличие инкапсуляции достаточно для объектности языка программирования, но ещё не означает его объектной ориентированности – для этого требуется наличие наследования.

Но даже наличие инкапсуляции и наследования не делает язык программирования в полной мере объектным с точки зрения ООП. Основные преимущества ООП проявляются только в том случае, когда в языке программирования реализован полиморфизм подтипов – возможность единообразно обрабатывать объекты с различной реализацией при условии наличия общего интерфейса.

Основные понятия ООП

Инкапсуляция – это свойство системы, позволяющее объединить данные и методы, работающие с ними, в классе, и скрыть детали реализации от пользователя.

Наследование – это свойство системы, позволяющее описать новый класс на основе уже существующего с частично или полностью заимствующейся функциональностью. Класс, от которого производится наследование, называется базовым, родительским или суперклассом. Новый класс – потомком, наследником, дочерним или производным классом.

Полиморфизм – это свойство системы использовать объекты с одинаковым интерфейсом без информации о типе и внутренней структуре объекта. При использовании термина «полиморфизм» в сообществе ООП подразумевается полиморфизм подтипов; а использование параметрического полиморфизма называют обобщённым программированием.

Среди многих распространённых программных продуктов, написанных на Delphi, можно найти:

• Продукция Embarcadero: Embarcadero Delphi, Embarcadero C++ Builder, Borland Jbuilder 1 и 2 версии.

• Администрирование и разработка баз данных: MySQL Tools, IBExpert.

• Инженерное программное обеспечение: Altium Designer, SprutCAM.

• Файловые менеджеры: Total Commander, Frigate, ViewFD.

• Просмотрщики графики: FastStone Image Viewer, FuturixImager, drComRead.

• Видео- и аудиопроигрыватели: Light Alloy, The KMPlayer, AIMP, X-Amp, Nata Player.

• Программы мгновенного обмена сообщениями: QIP 2012, R&Q, The Bat!, PopTray, FeedDemon.

• Клиенты файлообменных сетей: Shareman.

• Создание музыки: FL Studio, Guitar Pro (до версии 6.0).

• Разработка программного обеспечения: Dev-C++, Dunit, Game Maker, Inno Setup, PyScripter.

• Веб-разработка: Macromedia HomeSite.

• Текстовые редакторы: BirEdit, Notepad GNU, Bred.

• Бухучёт и налогообложение: «Бюджет 21», «Парус», AVARDA (до версии 6.x включительно).

• Программы для создания анимаций: Pivot Stickfigure Animator.

• Программы для сжатия данных: ALZip, PowerArchiver.

• Компьютерные игры: Age of Wonders, «Космические рейнджеры», Космические рейнджеры HD: Революция, Venom. Codename: Outbreak, Space Empires V, «Правда о девятой роте».

• Графические редакторы: Real Paint.

• Системные утилиты: Auslogics BoostSpeed.

• Системы управления очередью: МАКСИМА.

• Биллинговые системы: Петер-Сервис, Аргус.

• Спутниковое телевидение: DVBViewer.

1.3. Термин «Тест»

Термин «тест» начал применяться еще в старофранцузском языке. Этим словом

назывался сосуд небольших размеров, изготовленный из обожженной глины, который использовался алхимиками для проведения различных опытов.

Русский язык 19 века дает термину «тест» два значения:

испытательная присяга, английская религиозная клятва, которую произносил каждый вступающий в общественную должность как свидетельство того, что он не является тайным католиком;

плоский плавильный сосуд для выделения олова из золота или серебра.

В современном толковом словаре дается уже больше значений термину «тест»: проба, проверка, испытания, исследования. В связи с этим и область применения этого понятия значительно расширилась.

Тест (от слова англ. *test* – «испытание», «проверка»), тестирование – метод изучения глубинных процессов деятельности системы посредством помещения системы в разные ситуации и отслеживания доступных наблюдению изменений в ней.

Для нас тест – искусственно созданная ситуация, выбранная определённым образом, для проверки уровня знаний.

Было бы совершенно неправильно думать, что тесты можно использовать только для контроля знаний. Применение тестов в обучении – это одно из рациональных дополнений к методам проверки знаний, умений и навыков обучающихся, оптимально соответствующие процессу самостоятельной работы каждого ученика. Тесты индивидуализируют учебный процесс и реализуют одну из главнейших функций обучения – диагностирующую, которая позволяет обеспечить качественную обратную связь и своевременную коррекцию учебного процесса.

В последние годы наблюдается массовый переход к тестовым технологиям измерения качества обучения, позволяющим производить объективные оценки уровня знаний. Уровень подготовленности тестируемых является, к сожалению, параметром, недоступным для непосредственного измерения, чтобы «добраться» до него, необходимо использовать серьёзные научные методы составления качественных тестов и совместной обработки результатов тестирования. Качество любого теста оценивается известными характеристиками: объективность, надёжность, валидность, дискриминативность. Однако при проведении тестирования не всегда уделяется должное внимание выбору самых главных параметров теста: длине (количество заданий) и времени исполнения.

2. Создание программы на Delphi 7

2.1. Печатный вариант созданного теста

Для создания теста была выбрана тема «Тригонометрия» из курса «Алгебра и начала анализа» 10 класс, в которой необходимо проверить знания по следующим вопросам: тригонометрические функции, тригонометрические уравнения и преобразование тригонометрических выражений.

В части закрытого типа были продуманы следующие задания и варианты ответов:

Задание №1 – это задание с переводом градусов в радианы (рис. 5).

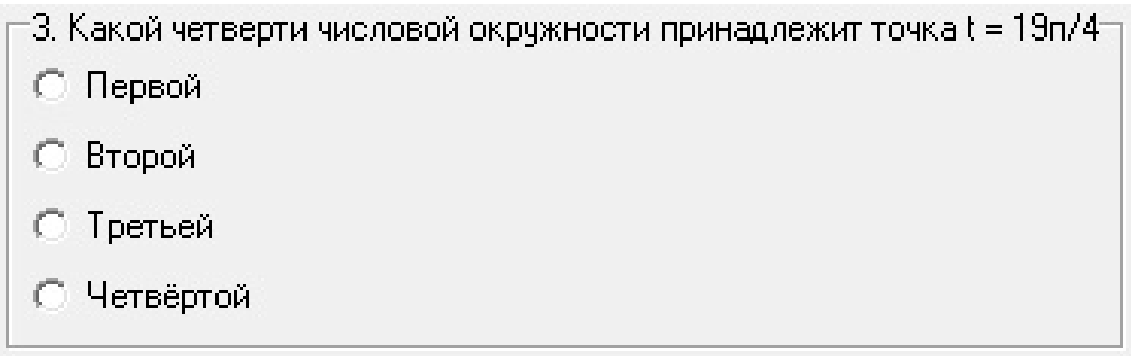

Рис. 5. Задание №1 из тестовой части

Задание №2 – это задание с переводом радиан в градусы (рис. 6).

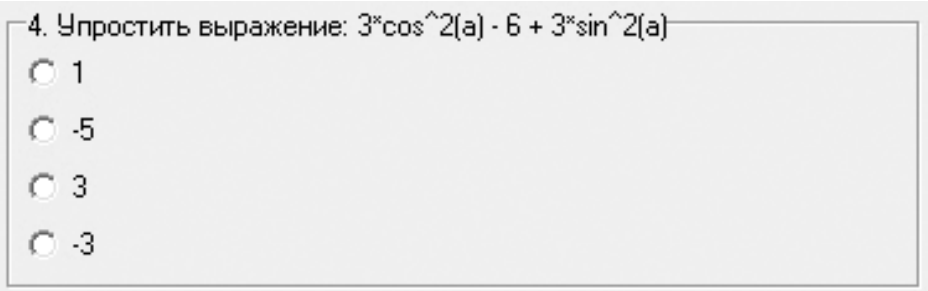

Рис. 6. Задание №2 из тестовой части

Задание №3 – это задание на определение положения точки на окружности.

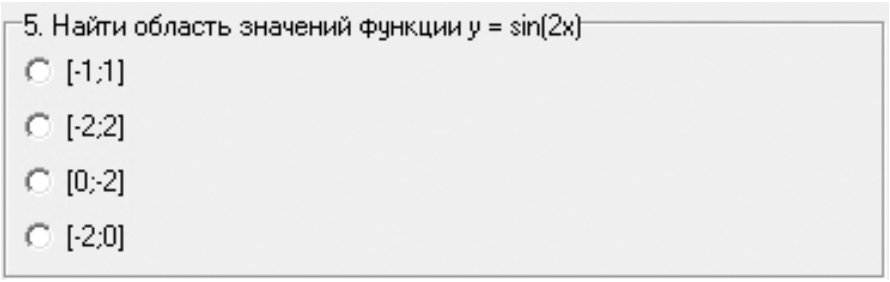

Рис. 7. Задание №3 из тестовой части

Задание №4 – это задание на упрощение выражения.

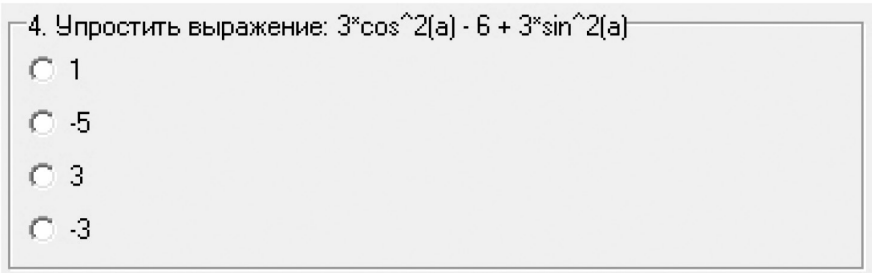

Рис. 8. Задание №4 из тестовой части

Задание №5 – это задание на поиск области значений функции.

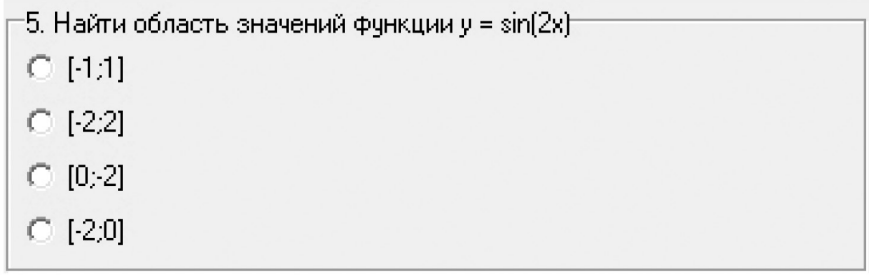

Рис. 9. Задание №5 из тестовой части

Остальные задания закрытой части теста подобны первым пяти и способствуют закреплению материала по пройденным темам.

Для открытой части заданий были продуманы следующие вопросы:

Задание №1 – это задание на преобразование графиков функций (рис. 10).

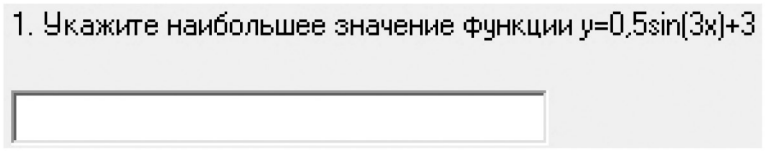

Рис. 10. Задание №1 из письменной части

Задание №4 – это задание на преобразование выражений (рис.11).

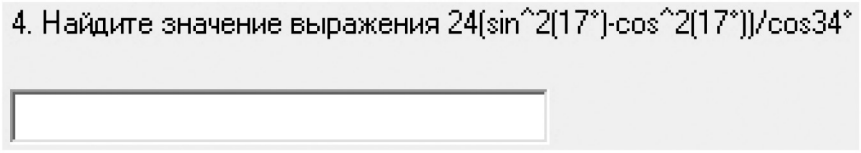

Рис. 11. Задание №4 из письменной части

Задание №6 – это задание для проверки знания формулы двойного угла (рис. 12).

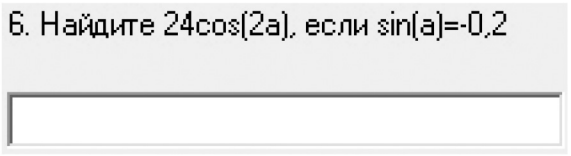

Рис.12. Задание №6 из письменной части

Задание №7 это задание на решение тригонометрических уравнений (рис.13).

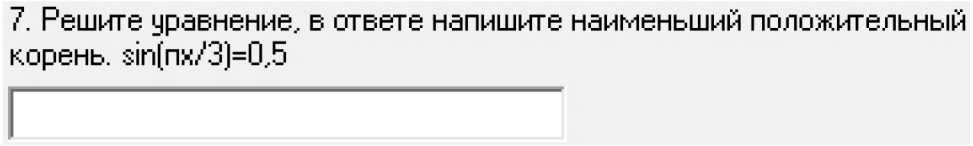

Рис. 13. Задание №7 из письменной части

Задание №10 – это задание на преобразование тригонометрических функций (рис. 14).

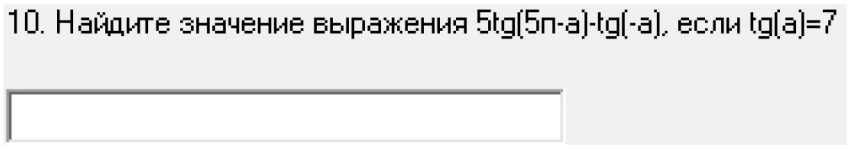

Рис. 14. Задание №10 из письменной части

Остальные задания письменной части подобны приведённым выше.

2.2. Пошаговая инструкция создания теста в Delphi 7

Общий вид теста и все функции будут перечислены и описаны ниже.

Так выглядит среда разработки в Delphi 7 (см. приложение 1, рис. 1).

Первым делом на нашей форме (Form1) создадим область, в которой будем работать, называется она **PageControl** в палитре **Win32** (см. приложение 1, рис. 2).

Далее создадим несколько новых страниц на **PageControl**. Для этого нужно нажать правой кнопкой мыши на эту область (см. приложение 1, рис. 3).

Необходимо дать название готовым вкладкам. Для этого в Инспекторе объектов (Object Inspector) в свойстве **Caption** введите название вкладки, у нас это «Инструкция» (см. приложение 1, рис. 4). Тем же способом даем название другой вкладки, у нас это «Тест». Таким образом можно создать любое количество вкладок.

Сейчас можно назвать нашу программу, для этого необходимо нажать на **Form1** в дереве объектов (Object TreeVeiew) у нас она будет называться «Тест». Название пишется в свойстве **Caption** (см. приложение 1, рис. 5).

На вкладке «Инструкция» (TabSheet1) создаём объект **Label** и задаём ему в Инспекторе объектов (Object Inspector) параметр **AutoSize** = False (для того чтобы ширина и высота поля **Label** были фиксированными). Текст **Label** пишется в свойстве **Caption**, у нас это «Инструкция» (см. приложение 1, рис.6). Выбрать шрифт текста можно, нажав на кнопку рядом с параметром **Font** (см. приложение 1, рис.7).

В следующем **Label** у нас будет инструкция по выполнению теста, установив те же параметры, что и у **Label1**, весь текст инструкции не влезет. Для этого внизу Инспектора объектов (Object Inspector) есть параметр **WordWarp** и его значение с **False** меняем на **True**, что позволит перемещать слова в следующую строку, когда закончится место (см. приложение 1, рис.8).

Естественно места для всех вопросов не хватит. Поэтому создаём объект **ScrollBox** (см. приложение 1, рис. 9), он создаст ползунок для перемещения между компонентами, которые будут в нём.

Наконец, создаем часть закрытого типа нашего теста, для этого в палитре **Standard** выбираем объект **RadioGroup**. Как и до этого основная надпись, в нашем случае сам вопрос, пишется в параметре **Caption** (см. приложение 1, рис. 10).

Варианты ответа на наш вопрос прописываются в параметре **Item** нашей **RadioGroup1**. Каждый ответ пишется в отдельной строке, количество кнопок под ответы обозначается цифрой перед словом **line**. По завершению просто нажать кнопку «Ок» (см. приложение 1, рис. 11). Количество вопросов и вариантов ответов не ограничено.

Переходим к открытой части нашего теста. Создаём ещё одну вкладку (**TabSheet3**) и назовём её, в нашем случае «Вопросы». В ней, так же, создаём объект **ScrollBox**. Следующий, новый для нас объект, **Edit**. **Edit** – поле для записи ответа, так же там можно написать свой вопрос (если это будет удобно) в параметре **Text** (см. приложение 1, рис. 12).

Теперь нужно создать кнопки, в которых будет прописан код. Они так же находятся на панели **Standard**. Создаём их внутри вкладки «Тест» (**TabSheet2**) и рядом с ними **Label** с теми же параметрами, что и второй. Подписать её так же нужно в строке параметра **Caption** (см. приложение 1, рис. 13). То же самое делаем для вкладки «Вопросы» (**TabSheet3**) (см. приложение 1, рис.14).

Также можно создать кнопку закрытия нашей программы. Хоть это и не обязательно, но всё-таки удобно, когда есть такая кнопка. Конечно, можно будет выйти из программы с помощью красного крестика в правом верхнем углу программы. Кнопка выхода называется **BitBtn** и находится она в палитре **Additional**, как всегда её название изменяется в поле параметра **Caption**. Важно создать её вне нашего **PageControl** (см. приложение 1, рис. 15), чтобы она отображалась на всех страницах. Далее её можно просто перенести (см. приложение 1, рис. 16).

Сейчас начинается самое интересное. Дважды нажав на кнопку проверки откроется окно Unit1.pas. Это наш код. Тут используется тот же самый код, что и в **Pascal**. Итак сам код должен выглядеть так (см. приложение 1, рис. 17). Если вопросов несколько, необходимо добавить нужное количество (по количеству **RadioGroup** в нашем тесте), у меня нас всего одна **RadioGroup** по этому индекс её «1» (**RadioGroup1**) и строка всего 1. На каждый отдельный вопрос пишется одна строка и меняется её индекс. То что написанно после «**ItemIndex=**» это номер варианта с правильным ответом. Нумерация вариантов начинается с 0, то есть если правильный ответ в первой строке, то мы должны написать «ItemIndex=0», если во второй «ItemIndex=1» соответственно и т.д.

Для второй кнопки код выглядит чуть чуть иначе, ведь там идёт вписывание ответа в **Edit**. Так же как и для первой кнопки нужно будет вписать нужное количество строк, изменяя индекс после слова «**edit**» (например Edit1) после «=», внутри апострофов записывается правильный ответ. Советую указать в инструкции как правильно записывать ответы в этих полях. Дальнейший код не отличается от предыдущего (см. приложение 1, рис. 18).

Если при тесте программы не работает кнопка выхода, то необходимо дважды нажать её на форме и прописать следующую комманду «close;» (см. приложение 1, рис. 19).

Осталось лишь чуть-чуть отредактировать оформление нашей программы. Как вы заметили, при тестах окно программы было размером с нашу исходную форму. Необходимо переместить нашу программу в левый верхний угол формы, кнопка **BitBtn** не привязана к **PageControl,** следовательно её нужно переместить отдельно. Далее просто «сузить» окно формы до размеров нашего **PageControl**. Изменим несколько параметров нашей формы: **biMaximize** установить на **False** это запретит пользователю разворачивать программу в полноэкранный режим. Так же **BorderStyle** установить на bsSingle. Последнее, что необходимо отметить, что нужно поднять все ползунки **ScrollBox** вверх и выйти на первую вкладку (**TabSheet1**), с которой собираетесь начинать работу (см. приложение 1, рис. 20).

Если все шаги выполнены правильно, у вас получится готовый тест, однако вам понадобится провести его апробацию.

Заключение

В целом программирование на Delphi существенно отличается от Pascal, но в части написания кода они идентичны. В Delphi можно визуально представить программу, редактировать её оформление, использовать компоненты из палитры, изменять функционал. С помощью Delphi можно быстро создавать готовый рабочий продукт.

Цели и задачи, поставленные в работе, выполнены. Освоен язык программирования Delphi 7, создан готовый продукт, разобрано понятие программирование и его языки, изучены виды языков программирования и потенциальные возможности применения Delphi, рассмотрена история языков программирования, дан ответ на вопрос «зачем в курсе десятого класса проходят программирование в Pascal?».

Итак, можно сделать вывод, что программирование зародилось достаточно давно, ещё до появления первых компьютеров, однако совершенствоваться оно начало лишь в период с 1960-х по 1970-х годов, что компиляторы и интерпретаторы относятся к языкам программирования низкого уровня, а Delphi и Pascal к языкам программирования высокого уровня. Delphi является императивным, структурированным, объектно-ориентированным языком программирования, диалектом Object Pascal. Что объектно-ориентированный подход в Delphi имеет инкапсуляцию, наследование и полиморфизм и что без них он бы не мог называться объектно-ориентированным. Термин «тест» имел множество значений, а сейчас его термин взят от английского слова «test», которое значит «испытание», «проверка». Также нельзя использовать тесты лишь для проверки уровня знаний, ведь тесты теперь – неотъемлемая часть обучения. И сейчас, тесты индивидуализируют учебный процесс и реализуют одну из главнейших функций обучения – диагностирующую, которая позволяет обеспечить качественную обратную связь и своевременную коррекцию учебного процесса. И что качество любого теста оценивается известными характеристиками: объективность, надёжность, валидность, дискриминативность. Однако при проведении тестирования не всегда уделяется должное внимание выбору самых главных параметров теста: длине (количество заданий) и времени исполнения.

Также мы разобрали примеры заданий созданного теста по тригонометрии. Разработали инструкцию по созданию своей программы в Delphi 7, которая, при верном исполнении, даст нам готовый тест по любому, выбранному предмету, в любой области. И будет давать объективную оценку знаний учащегося.

Во время написания программы возникла трудность с кодированием верных ответов, а конкретно необходимо было заменить предлагаемую систему массивов на более простую развилку, иначе выдавались случайные числа в параметре «+IntToStr(ball)+».

Тест был апробирован на старшеклассниках образовательных учреждений города Перми. Во время апробации были выявлены многие разнотипные ошибки, которые впоследствии были исправлены. От одного из участников апробации был получен похвальный отзыв: «Программа очень удобна. В тесте присутствуют интересные задания, которые отсутствовали в школьной программе. Сложные, но решаемые, задания очень понравились. Достаточно удобная форма тестирования, понятные инструкции по выполнению заданий. Интересное пользование программой. В общем, остался доволен. Не присутствуют отвлекающие факторы, всё просто и лаконично. Андрей, 17 лет».

Написав эту работу, я окончательно решил проблему с выбором профессии. Осознал необходимость программирования тестов для учащихся, так же я помог своим одноклассникам перед контрольной работой по теме «Тригонометрия», благодаря чему 55% учащихся 10 «А» класса получили отметки «хорошо» и «отлично». Но известно, что в Delphi можно создавать не только тесты, я смогу создавать любые программы по заданному алгоритму.

Список литературы

1. Грэхем И. Объектно-ориентированные методы. Принципы и практика / Иан Грэхем. – М.: Вильямс, 2004. – С. 880.

2. Иванов А.А., Иванов А.П. Тематические тесты для систематизации знаний по математике. Ч. 1. – Пермь: Изд-во Пермского университета, 2006. – 210 с.

3. Половина И.П. Лабораторный практикум по Delphi / Е.А. Еремин. – Пермь: Изд-во Пермского регионального института педагогических информационных технологий, 1999. – 84 с.

4. Пышкин Е.В. Структурное проектирование: основание и развитие методов. С примерами на языке С++. / Е.В. Пышкин — СПб.: Политехнический университет, 2005. – 324 с.

5. Себеста Р.В. Основные концепции языков программирования. / Роберт В Себеста. – М.: «Вильямс», 2001. – 672 с.

6. Программирование в среде Delphi [Электронный ресурс]. – Режим доступа: http://ru.wikipedia/Программирование. (10.02.2015).

7. Языки программирования [Электронный ресурс]. – Режим доступа: http://ru.wikipedia/Язык_программирования. (1.02.2015).